Fakulta Matematiky, fyziky a informatiky Univerzity Komenského


# Metodika vývoja multi-agentových aplikácií

Autor: Peter Žiak
Dipl. vedúci: Mgr. Radovan Červenka

Bratislava
Apríl 2005

Čestne prehlasujem, že na predkladanej diplomovej práci som pracoval samostatne a použil som literatúru uvedenú na konci práce.

…….

Peter Žiak

Rád by som vyjadril svoje poďakovanie vedúcemu diplomovej práce Mgr. Radovanovi Červenkovi za podnetnú diskusiu a cenné pripomienky pri písaní tejto diplomovej práce, rovnako ako aj svojej rodine za podporu pri jej písaní.

# 1 Úvod

## 1.1 Motivácia

Agentová platforma a súvisiace technológie majú v poslednej dobe tendenciu stať sa plnohodnotnou disciplínou softvérového inžinierstva a sú snahy presadiť multi-agentové systémy aj priemyselne. Existujú aktivity v oblasti teórie a výskumu, sú snahy o štandardizáciu, existuje niekoľko implementácií agentových platforiem a v niektorých oblastiach sa začínajú úspešne uplatňovať aj agentové architektúry.

Neoddeliteľnou súčasťou priemyselného uplatnenia agentových technológií je aj existencia procesu vývoja agentových systémov. V súčasnej dobe existuje niekoľko formálnych aj neformálnych návrhov metodík vývoja agentových aplikácií. Tieto majú určité spoločné črty ale aj odlišnosti. Niektoré z nich majú tendenciu byť univerzálne, iné sú určené pre špecifické agentové platformy a technológie. Problémom je, že neexistuje všeobecne akceptovaná a generická metodika alebo koherentný súbor metodík. Dôvodom neustáleho vývoja metodík softvérového inžinierstva je tvorba metodík zabezpečujúcich proces vývoja kvalitného softvéru pri maximálnom minimalizovaní nákladov. Rozhodnúť, použitie ktorej metodiky je správne, je veľmi náročné. A takéto rozhodnutie je ešte náročnejšie v období vyvíjajúcich sa technológií, akými sú dnes aj multi-agentové systémy.

### 1.1.1 História

V súčasnosti si môžeme všimnúť, že sa agentovo-orientované programovanie ako nová paradigma softvérového inžinierstva presadzuje viac a viac. Dôvodom je, že softvérový priemysel má tendenciu čoraz častejšie vytvárať väčšie a komplexnejšie systémy. Napriek tomu veľa ľudí z IT nemá o agentoch takmer žiadne vedomosti a pri spomenutí agentov v rozhovore reagujú: „Agent? Kto/čo to je?“ V tejto práci sa zameriavame na metodiky vývoja multi-agentových aplikácií a nebudeme venovať pozornosť konkrétnym architektúram alebo modelovacím jazykom.

### 1.1.2 Agent – čo/kto to je?

Agent je v súčasnosti módnym slovom v odbornej i populárno-vedeckej literatúre a tak niet divu, že doteraz neexistuje jediná, všeobecne uznaná a platná

definícia, ktorá by pokrývala všetky snahy a výsledky v oblastiach agentovo orientovaného programovania a multi-agentových systémov. Väčšina autorov nových systémov uvádza aj definíciu agenta, ale často s dôrazom na špecifické autorovo vnímanie vlastného systému, pričom väčšinou unikajú resp. ustupujú do úzadia všeobecne platné vlastnosti, ktoré by agent mal mať.

Napriek tomu asi najčastejšie citovanou definíciou je nasledujúca, pochádzajúca od autorov Woolbridga a Jenningsa [01], ktorá rozlišuje medzi tzv. slabým a silným agentom, pričom slabého agenta definujú ako „počítačový systém s nasledujúcimi vlastnosťami:

- autonómnosť - agenty pracujú bez priamych ľudských alebo iných zásahov a majú určitým spôsobom kontrolu nad svojim konaním a vnútorným stavom,
- sociálna schopnosť - agenty spolupracujú s inými agentmi (prípadne ľuďmi) prostredníctvom určitého druhu komunikačného jazyka agentov,
- reaktivita - agenty vnímajú svoje okolie a vo vhodnom čase reagujú na zmeny vo svojom okolí,
- pro-aktivita - konanie agentov nie je len jednoduchou odozvou na stav okolia, ale môže byť aj s prevzatím iniciatívy.“

Pri definícii silného agenta autori pridávajú k týmto vlastnostiam, napríklad v disciplíne implementácie, aj potrebu použitia pojmov charakteristických pre ľudí, akými sú napr. pojmy charakterizujúce duševné a emocionálne stavy (vedomosť, viera, zámer, záväzok, ...).

V článku [02] autori Franklin a Graesser uvádzajú vlastnú definíciu, ktorá poukazuje na odlišnosť programových agentov od bežných programov:
„Autonómny agent je systém, ktorý je časťou prostredia, ktoré vníma, na ktoré pôsobí a v ktorom existuje v priebehu času, pričom vykonáva vlastné úlohy a tak ovplyvňuje to, čo bude vnímať v budúcnosti.“ V tejto definícii autori poukázali na schopnosť agenta meniť svoje vnímanie prostredia plynutím času, a teda jeho odlišnosť jednorazového vykonania programu [03].

### 1.1.3 Multi-agentové systémy

Multi-agentové systémy (MAS) je oblasť softvérové inžinierstva zaoberajúca sa skúmaním možnej koordinácie viacerých podsystémov s cieľom dosiahnuť vyššie schopnosti, teda schopnosti, ktoré nevlastní a ktorých nie je schopný ani jeden zo subsystémov individuálne.

## 1.2 Ciele

V tejto práci sme sa zamerali na zozbieranie a preštudovanie existujúcich metodík, odporúčaní, postupov a techník tvorby multi-agentových systémov. Priniesli sme ich popis a ich vzájomné porovnanie. Na základe porovnania existujúcich metodík, získaných znalostí a skúseností, je súčasťou práce aj odporúčaný postup vývoja, ktorý by sa mal uplatniť v generickej metodike vývoja multi-agentových systémov a vytvorenie „kostry“ takejto metodiky.

## 1.3 Štruktúra práce

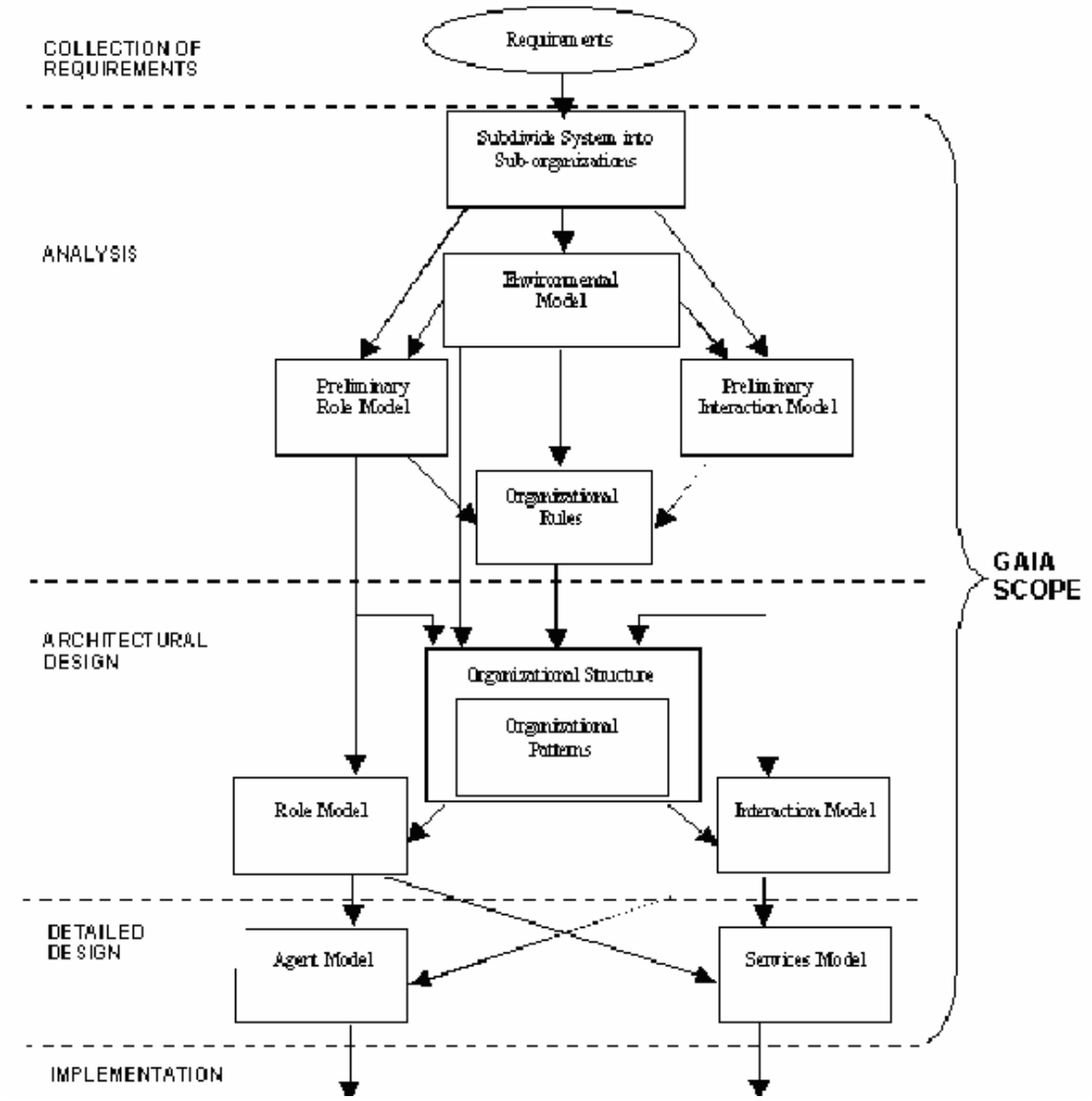
Stručne si teraz priblížime organizáciu tejto práce. Na úvod uvedieme základné poznatky o multi-agentových systémoch. Popíšeme existujúce metodiky a následne ich vzájomne porovnáme, identifikujeme ich dôležité časti, zhodnotíme výhody a nevýhody jednotlivých metodík a načrtneme vlastnosti všeobecnej metodiky vývoja multi-agentových systémov. V závere ešte ukážeme mapovanie niektorých existujúcich metodík na nami navrhnutú všeobecnú metodiku.

# 2 Prehľad existujúcich metodík

V druhej kapitole uvedieme niekoľko existujúcich metodík vývoja MAS. Každá z nich používa svoje vlastné pojmy a frázy, preto v prípade nezrovnalostí sa obráťte na použitú literatúru uvedenú na konci práce. My sme sa snažili použiť terminológiu používanú v meta-modelovacom jazyku SPEM.

## 2.1 Metodika Gaia

Proces metodiky Gaia[04][05][06] pozostáva z troch disciplín: disciplíny zhromažďovania požiadaviek, z disciplíny analýzy a z disciplíny dizajnu. Každá disciplína vytvára dokument skladajúci sa obvykle zo zoskupenia UML modelov, produktov činností a popisov práce. Obrázok č. 1 zobrazuje priebeh procesu metodiky Gaia [07][08].




*Obr. č.1: Modely metodiky Gaia a ich vzťahy v rámci Gaia procesu [08]*

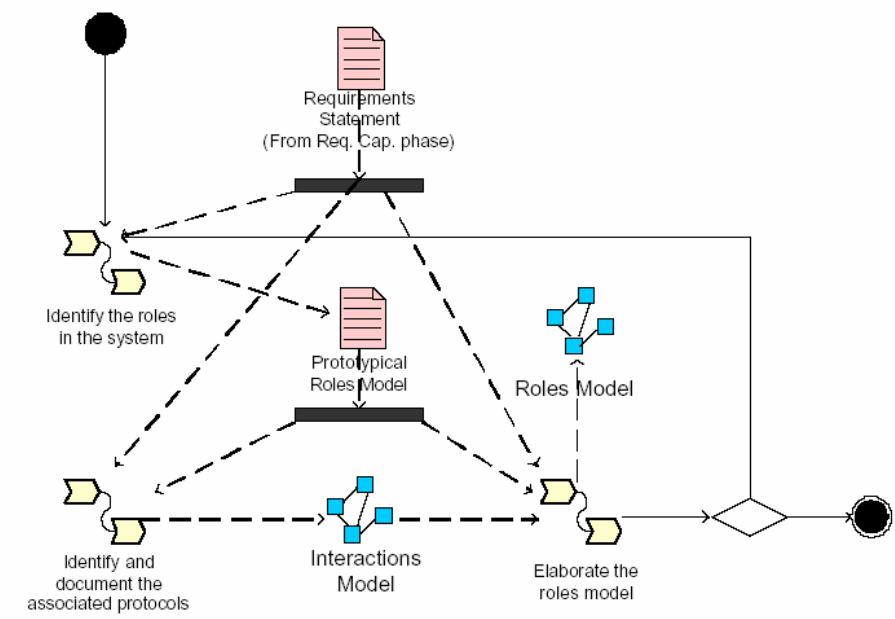
### 2.1.1 Disciplína zhromažďovanie požiadaviek:

V disciplíne zhromažďovanie požiadaviek je dôležité identifikovať základné schopnosti, funkčnosti a kompetencie, požadované organizáciou. Taktiež je dôležité identifikovať základné interakcie, ktoré sú potrebné na zužitkovanie identifikovaných schopností. Metodika Gaia neobsahuje popis procesu, ktorým by sa mal analytik riadiť za účelom získania dokumentu špecifikovaných požiadaviek. Rovnako neobsahuje ani popis samotného dokumentu špecifikovaných požiadaviek. Pri vstupe do nasledujúcej disciplíny, disciplíny analýzy, sa však predpokladá existencia tohto dokumentu.

### 2.1.2 Disciplína analýza:

Zámer disciplíny analýzy je pochopenie systému a jeho štruktúry v zmysle rolí a interakcií medzi rôznymi rolami.

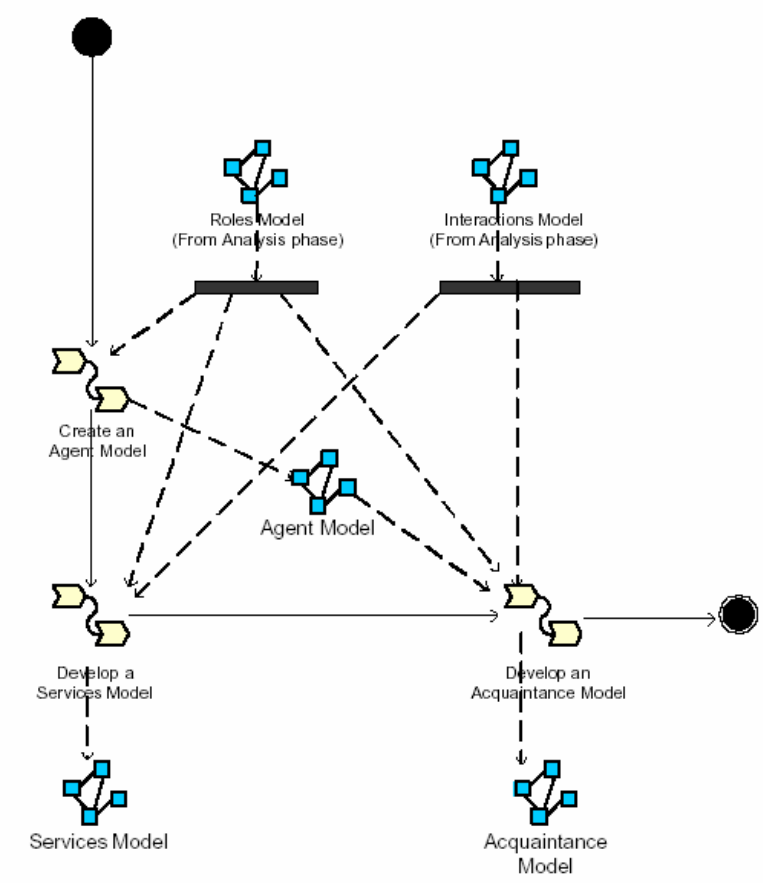
Prvý krok pozostáva z identifikovania rolí vytváraného systému a vytvorenia prvotného modelu rolí, pozostávajúceho zo zoznamu kľúčových rolí v systéme a ich popisu. Analytik následne pre každú rolu identifikuje a zdokumentuje asociované protokoly pomocou modelu interakcií. Protokoly sú vzory interakcií medzi rôznymi rolami v rámci systému a každý z nich je charakterizovaný atribútmi: zámer, iniciátor, kto odpovedá, vstupy, výstupy, spracovanie. Vychádzajúc z modelu interakcií systémový analytik vypracuje model rolí, v ktorom sú zdokumentované kľúčové role systému a každá z rolí je definovaná protokolmi, aktivitami, oprávneniami a povinnosťami.




*Obr. č.2: Disciplína analýzy metodiky Gaia vyjadrená v SPEM[09]*

### 2.1.3 Disciplína dizajn:

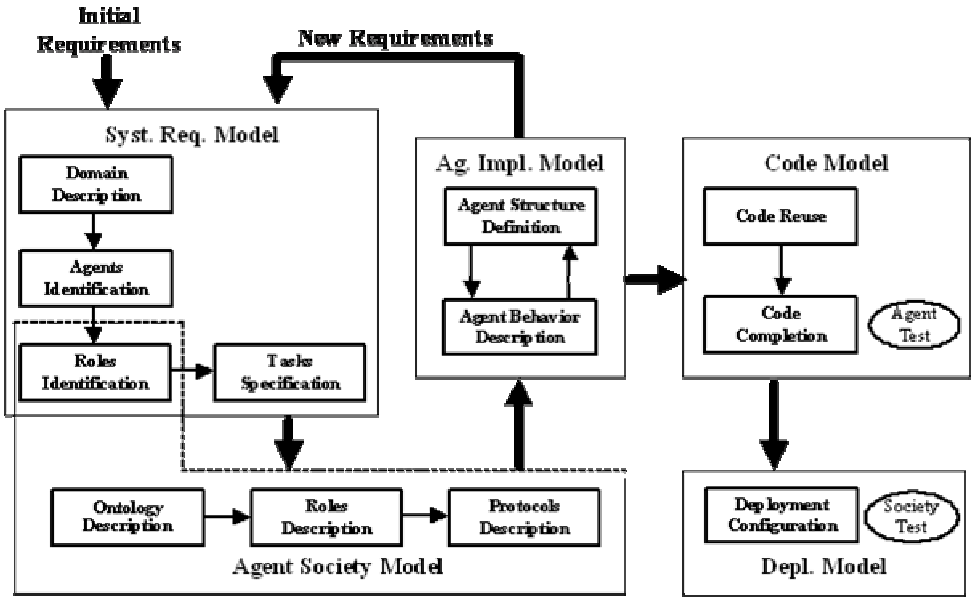
Disciplína dizajn zahŕňa vytvorenie troch modelov: Agentového Modelu, Modelu Služieb, Modelu okruhu známych. Agentový dizajnér je zodpovedný za: zoskupenie rolí do agentových typov, zdokumentovanie každého agentového typu, identifikovanie služieb a identifikovanie vzájomných vzťahov okruhu známych [08].




*Obr. č. 3: Disciplína dizajnu metodiky Gaia*

Agentový model identifikuje typy agentov tvoriacich systém a agentové inštancie, ktoré budú inštanciované z týchto typov. Model služieb identifikuje služby asociované s každou agentovou rolou. Zároveň je potrebné identifikovať vstupy a výstupy (z modelu interakcií) a vstupné a výstupné podmienky každej služby. Model okruhu známych dokumentuje komunikačné spojenia medzi rôznymi agentmi. Zámerom tohto modelu je identifikovanie možných komunikačných úzkych miest a je ho možné reprezentovať ako orientovaný graf, ktorý je možné získať priamo z agentového modelu a modelov rolí a interakcií. Disciplínu dizajnu metodiky Gaia je vidieť na obrázku č. 3.

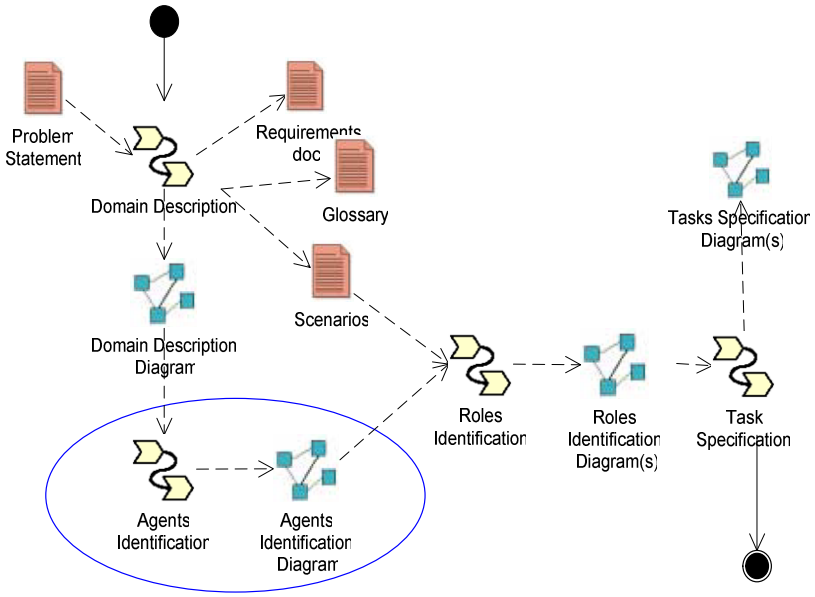
## 2.2 Metodika PASSI




*Obr. č. 4: Kompletný proces metodiky PASSI*

Metodika PASSI (Proces for Agent Societies Specification and Implementation) [10][11] je tvorená viacerými modelmi ako je vidieť na obrázku č. 4:

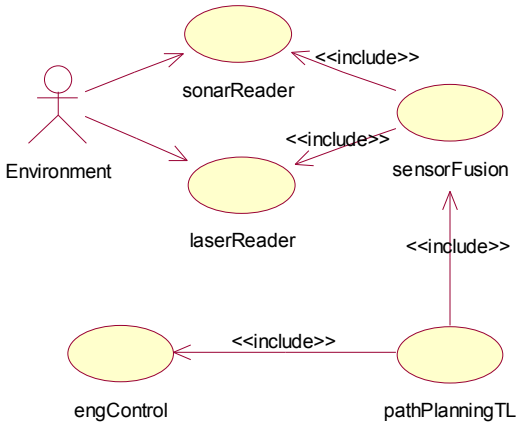
### 2.2.1 Model systémových požiadaviek




*Obr. č. 5: Model systémových požiadaviek metodiky PASSI vyjadrený cez SPEM*

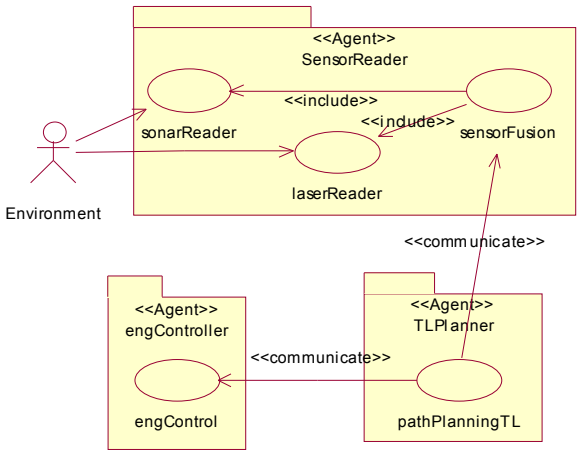
Vytvorenie modelu systémových požiadaviek pozostáva zo štyroch fragmentov:

a) Popis domény: Cieľom tohto fragmentu je pomocou use-case diagramu/ov popisu domény a pomocou textového dokumentu požiadaviek systému zachytiť funkčný popis systému. Dokument požiadaviek systému obsahuje dokumentáciu use-case-ov: názov, zúčastnených aktorov, vstupné podmienky, tok udalostí, výstupné podmienky, výnimky, a špeciálne podmienky.



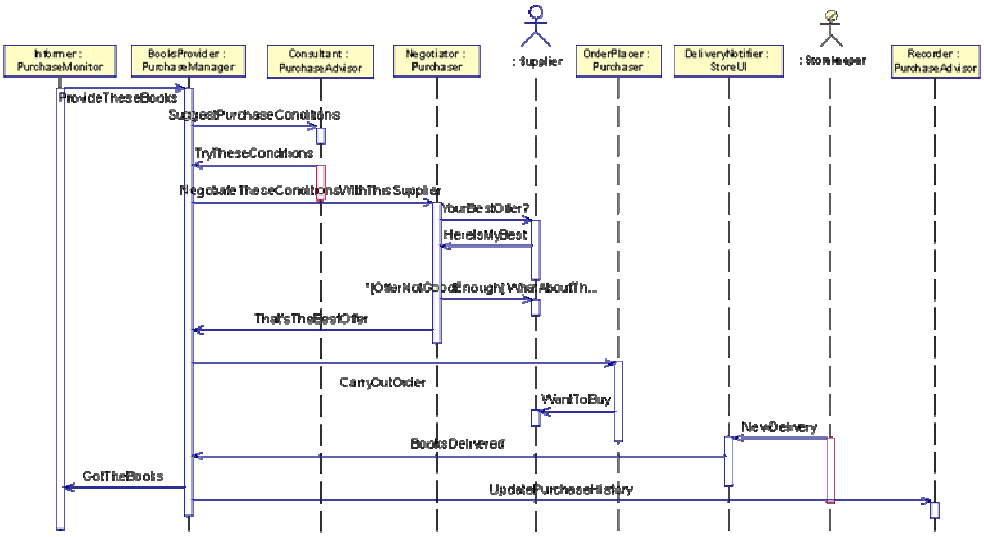

*Obr. č. 6: Príklad modelu popisu domény.*

b) Identifikácia agentov: Systémový analytik sa zaoberá identifikovaním všetkých agentov zainteresovaných v systéme a externých entít interagujúcich so systémom. Externé entity sú reprezentované ako aktori. Agent sa na základe svojich lokálnych vedomostí a schopností snaží dosiahnuť svoje ciele. Každý agent môže požiadať ďalších o spoluprácu, pokiaľ to neodporuje ich vlastným cieľom. Interakcie medzi agentmi a aktormi, podobne ako medzi rôznymi agentmi, pozostávajú z komunikácií. Agent môže svoje vedomosti rozširovať komunikovaním s inými agentmi alebo skúmaním reálneho sveta.




*Obr. č. 7: Príklad identifikovaných agentov a entít*

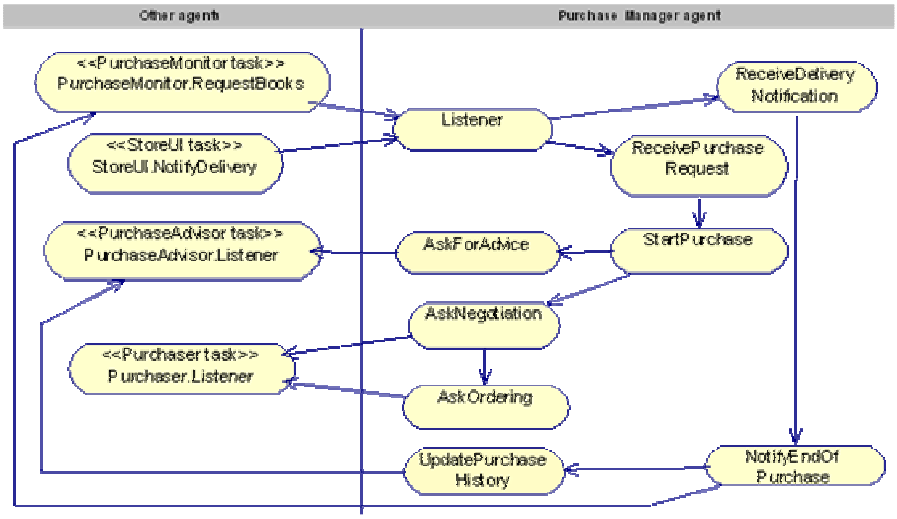
c) Identifikácia rolí: Identifikácia rolí pozostáva z vytvorenia sekvenčného diagramu popisujúceho všetky komunikačné cesty medzi agentami, pričom cesta popisuje scenár interagujúcich agentov pracujúcich s cieľom dosiahnuť požadované správanie systému. Rola je zoskupenie úloh vykonávaných agentom za účelom dosiahnutia cieľov/podcieľov. Rola zároveň popisuje nejaký aspekt životného cyklu agenta a je spojená s agentovým poskytovaním služby pre spoločenstvo ostatných agentov za účelom dosiahnutia svojich alebo ostatných cieľov.




*Obr. č. 8: Príklad identifikácie rolí a tried - meno každej triedy tvorí dvojica (meno role: meno agenta)*

d) Špecifikácia úloh: Fragment špecifikácia úloh popisuje schopností každého agenta pomocou diagramov aktivít. Pre každého agenta je potrebné vytvoriť diagram aktivít, ktorý agent môže používať pre svoje úlohy na vykonávanie svojich plánov. Diagram

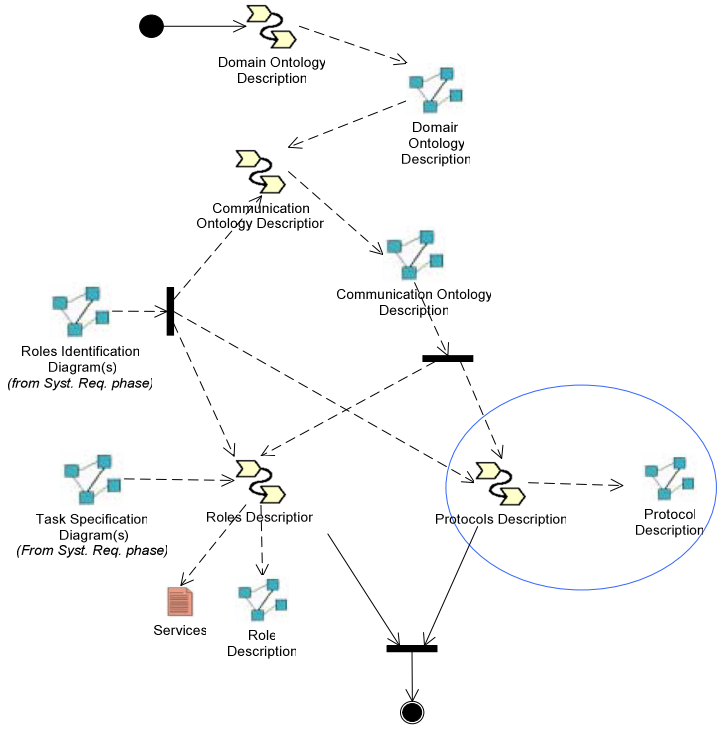
pozostáva z dvoch pruhov – v jednom (ľavom) sú úlohy ostatných agentov, ktorí interagujú s tým pre ktorého je diagram kreslený, a v druhom (pravom) sú len jeho vlastné úlohy. Prepojenia medzi pruhmi reprezentujú konverzáciu medzi agentmi a prepojenia v rámci pravého pruhu reprezentujú tok riadenia medzi rôznymi úlohami jedného agenta.




*Obr. č. 9: Príklad špecifikovania úloh agentov.*

### 2.2.2 Model spoločenstva agentov

je model sociálnych interakcií a závislostí medzi agentmi zahrnutých v riešení pomocou ich rolí, sociálnych interakcií, závislostí a ontológie.



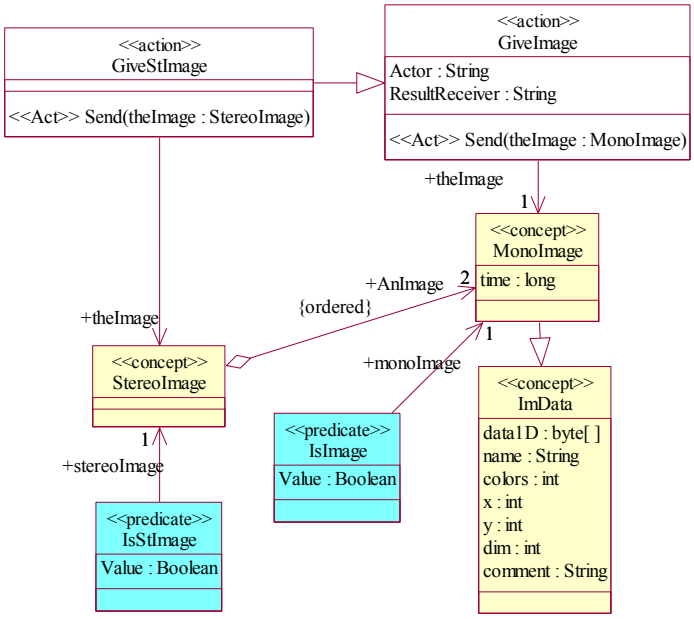

*Obr. č. 10: Model spoločenstva agentov*

Vytvorenie tohto modelu zahŕňa štyri kroky zahŕňajúc časť predchádzajúceho modelu:

a) Identifikácia rolí: Je rovnaká aktivita ako v modeli systémových požiadaviek.

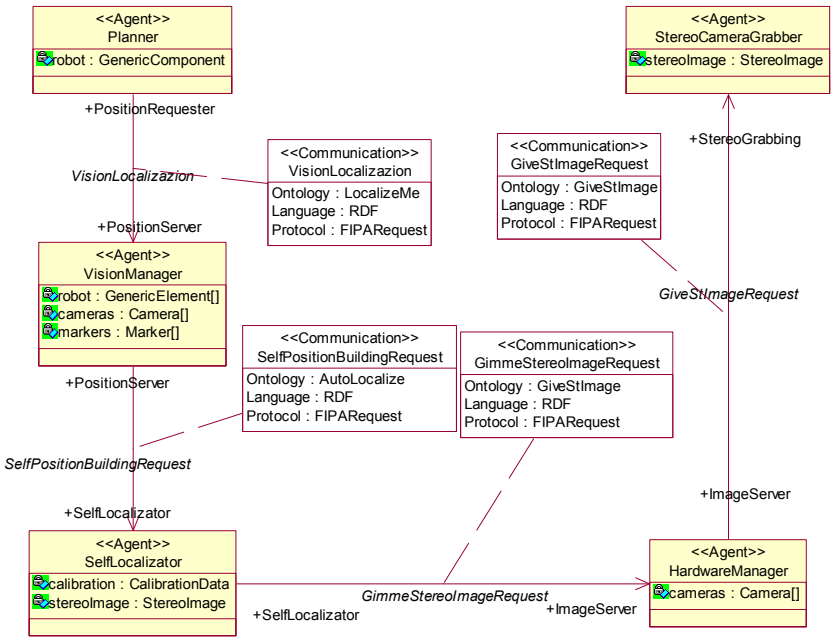
b) Popis ontológie: Popis ontológie slúži na popis vedomostí pripisovaných individuálnym agentom a zmyslom ich interakcií a pozostáva z dvoch fragmentov:

b1) Popis ontológie domény: Cieľom tohto fragmentu je navrhnutie ontológie systému. Ontológia sa popisuje diagramami tried pomocou konceptov (žltá farba), predikátov (modrá farba) a akcií (biela farba). Tieto elementy môžu byť vzájomne previazané pomocou troch štandardných UML vzťahov: generalizácia, asociácia, agregácia.



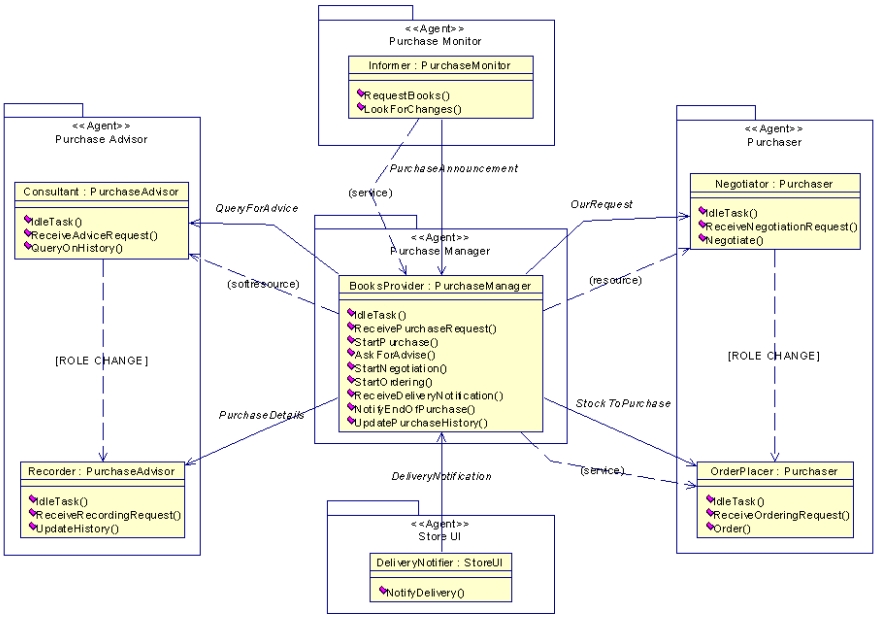

*Obr. č. 11: Príklad popisu ontológie domény*

b2) Popis ontológie komunikácií: Tento fragment sa zaoberá modelovaním sociálnych interakcií a závislostí medzi agentmi podieľajúcich sa na riešení. Diagram ontológie komunikácií je diagram tried pozostávajúci hlavne z agentov a komunikácií. Každý agent (žltá farba) je popísaný pomocou svojich vedomostí, ktoré predstavujú časti ontológie popísanej v predchádzajúcom diagrame. Každá komunikácia (biela farba) je reprezentovaná vzťahom medzi dvoma agentmi a detailnejšie informácie sú zachytené v atribútoch triedy. Trieda je identifikovaná jedinečným názvom a je popísaná pomocou atribútov: ontológia, jazyk a protokol.



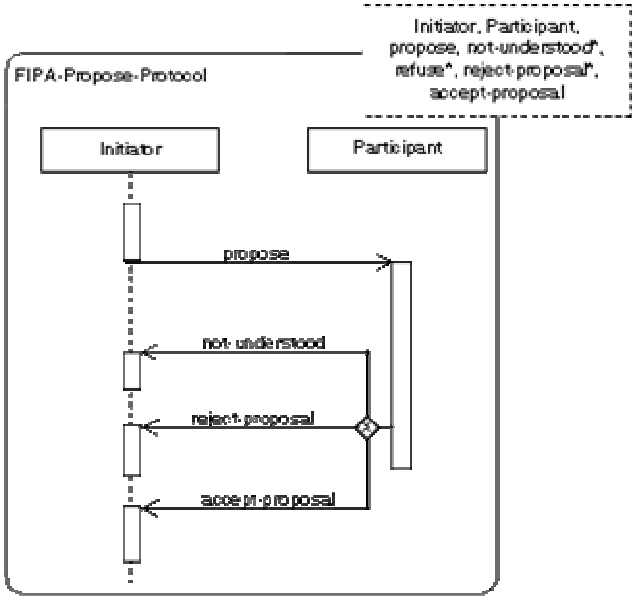

*Obr. č. 12: Príklad popisu ontológie komunikácií*

c) Popis rolí: Cieľom aktivity popis rolí' je vytvoriť model životného cyklu každého agenta, zohľadňujúc pritom role v ktorých môže vystupovať a potreby spolupráce a komunikácie s ostatnými agentmi. Na zobrazenie jednoznačných rolí hraných agentami, úloh zahrnutých v roliach, komunikačných schopností a závislostí medzi agentami používa diagramy tried, ktoré sú pre každého agenta samostatne zoskupené do balíkovského diagramu.



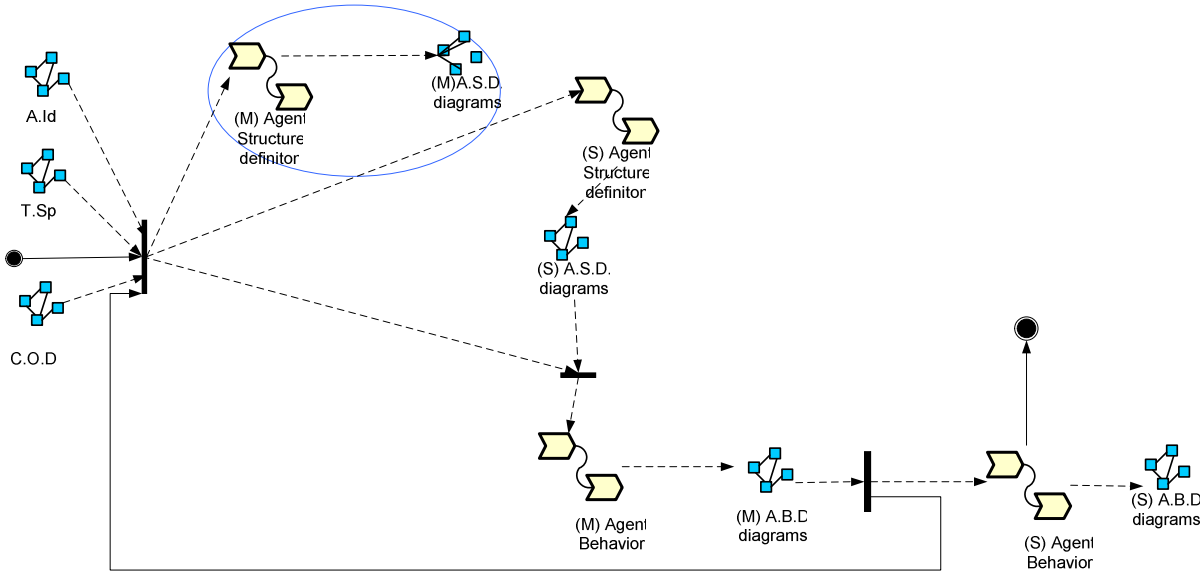

*Obr. č. 13: Príklad popisu rolí.*

d) Popis protokolov: Fragment popis protokolov slúži na špecifikovanie gramatiky každého potrebného komunikačného protokolu a na reprezentáciu používa sekvenčné diagramy. Sekvenčný AUML diagram potrebné vytvoriť pre každý neštandardný protokol. Príklad sekvenčného AUML diagramu je na obrázku nižšie.




*Obr. č. 14: Príklad popisu protokolov.*

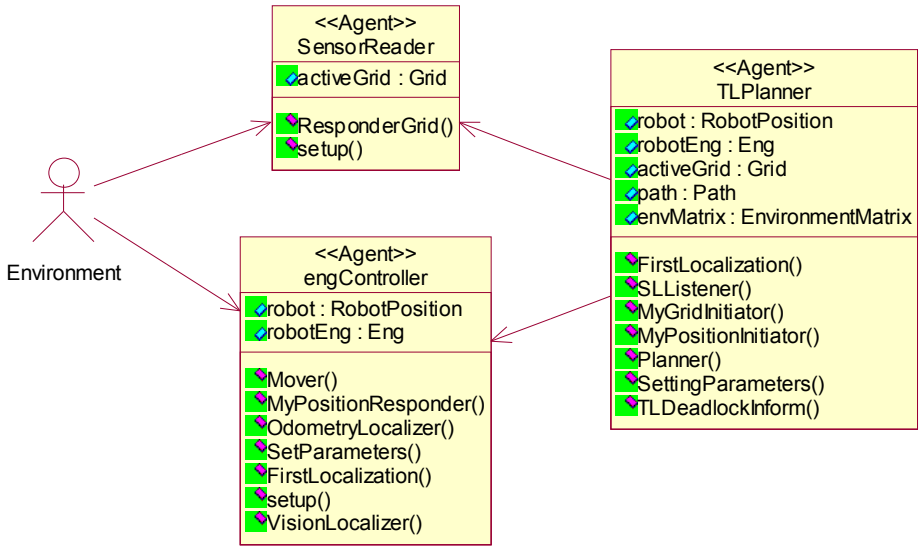
### 2.2.3 Model implementácie agentov.




*Obr. č. 15: Model implementácie agentov v metodike PASSI.*

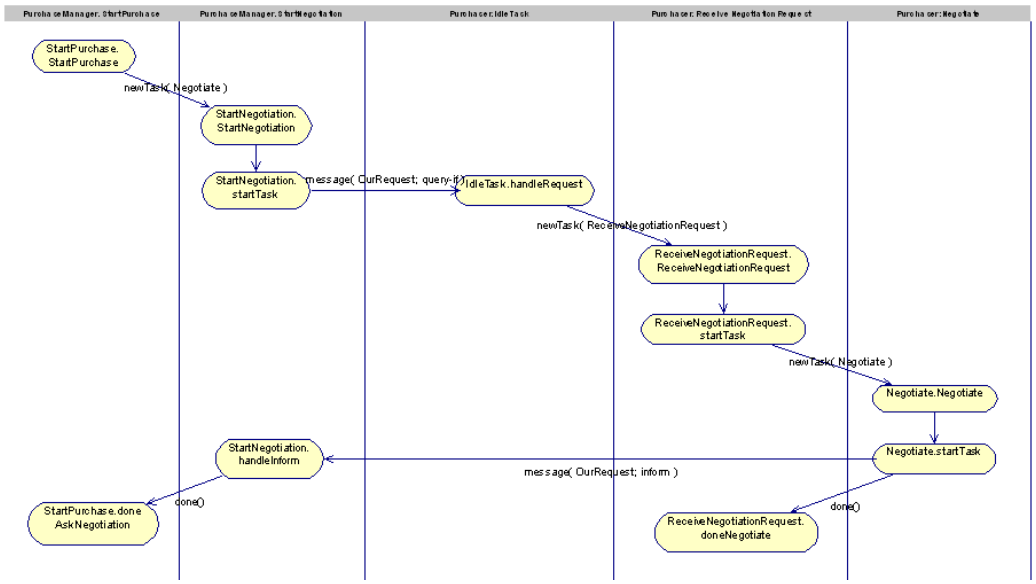
Model riešenia architektúry pomocou tried a metód, vývoj ktorých zahŕňa nasledovné fragmenty:

a) Definícia štruktúry agentov: Používa diagramy tried na popis štruktúry riešenia agentových tried. Diagram tried obsahuje aktorov a triedy (každá trieda reprezentuje jedného agenta systému). Atribúty v diagrame triedy sú použité na reprezentovanie vedomostí agenta (odkazujú na entity definované v popise domény ontológie) a operácie v diagrame triedy sú použité na naznačenie agentových úloh. Relácie medzi agentmi predstavujú tok komunikácií (výmeny informácií)



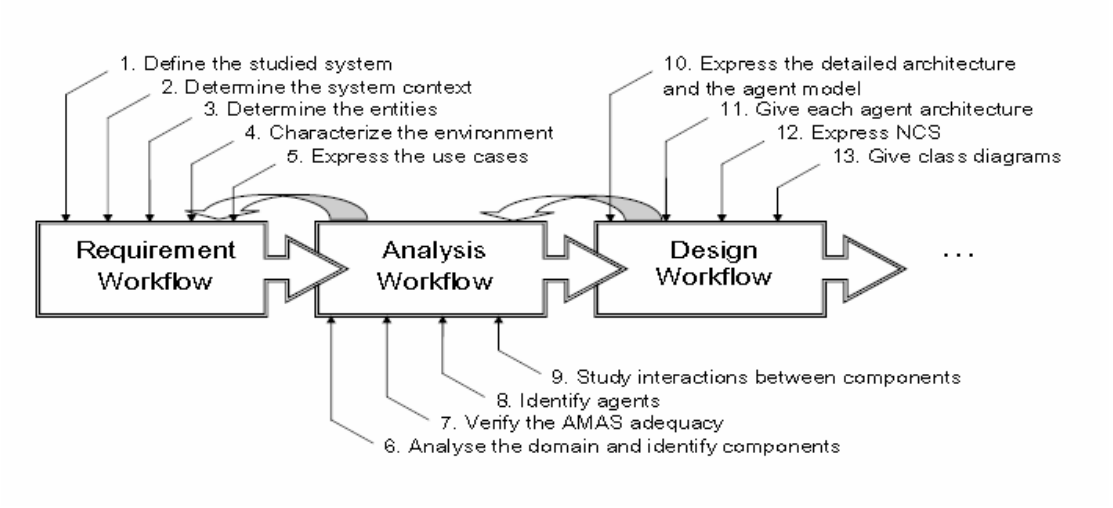

*Obr. č. 16: Príklad diagramu definície štruktúry agentov.*

b) Popis správania agentov: používa diagramy aktivít alebo stavové diagramy na popis správania jednotlivých agentov. Úlohou tohto fragmentu je zobrazenie toku udalostí medzi hlavnými agentovými triedami, rovnako ako aj medzi ich podtriedami, ktoré reprezentujú ich úlohy. Pre každého agenta a pre každú jeho úlohu je použitý jeden pruh. Aktivity v rámci pruhov predstavujú metódy súvisiacich tried.




*Obr. č. 17: Príklad diagramu popisu správania agentov.*

## 2.3 Metodika ADELFE (Atelier de Développement de Logiciels à Fonctionnalité Emergente)




*Obr. č. 18: Proces metodiky ADELFE.*

Metodika ADELFE[13][14][15] poskytuje v každej svojej aktivite možnosť vrátenia sa k niektorej už skôr vykonanej aktivite.

### 2.3.1 Disciplína skorých požiadaviek

**Aktivita 1: Definovanie užívateľových požiadaviek:** Táto aktivita sa zameriava na popis systému a prostredia, v ktorom je systém vyvíjaný. Zákazníci, koncoví užívatelia, analytici a dizajnéri musia spolu vytvoriť zoznam možných požiadaviek. Pomocou zoznamu definujú, aký systém vytvoriť, a aký je najvhodnejší pre koncových užívateľov. Zistené funkčné a nie-funkčné požiadavky sú spísané do dokumentu požiadaviek. Dokument požiadaviek môže byť jednoduchý textový dokument. Výstupom tejto aktivity je prvotná verzia dokumentu požiadaviek.

**Aktivita 2: Validácia užívateľových požiadaviek:** Koncový užívateľ overí akceptovateľnosť a platnosť posledného dokumentu požiadaviek vytvoreného analytikom. Ak nie je s dokumentom spokojný, dokument je vrátený do predchádzajúcej aktivity na doplnenie.

**Aktivita 3: Definícia požiadaviek odsúhlasených oboma stranami:** Dokument požiadaviek sa odovzdá na odsúhlasenie analytikovi a koncovým užívateľom. Ak nie je dokument odsúhlasený oboma stranami, je vrátený späť do predchádzajúcej aktivity a reorganizovaný.

**Aktivita 4: Vytvorenie množiny kľúčových slov:** Analytik extrahuje kľúčové slová, popisujúce systém a prostredie, z dokumentu požiadaviek do dokumentu kľúčových slov. Túto aktivitu je možné vykonať paralelne s nasledujúcou aktivitou.

**Aktivita 5: Extrakcia limitov a obmedzení:** Analytik rozšíri dokument požiadaviek o obmedzenia systému podľa obmedzení operačného systému, jazykov a technológie. Poslúžia mu pritom nie-funkčné požiadavky získané v aktivite 1 pri definovaní užívateľových požiadaviek.

### 2.3.2 Disciplína neskorých požiadaviek

**Aktivita 6: Charakterizovanie prostredia:** Hlavným cieľom tejto aktivity je definovanie prostredia systému v dokumente definície prostredia. Pozostáva z troch krokov:

1. zisti entity – analytik musí identifikovať aktívne a pasívne entity interagujúce so systémom a obmedzenia týchto interakcií.
2. definuj súvislosti – analytik charakterizuje toky údajov a interakcie medzi entitami a systémom. Toky údajov medzi pasívnymi entitami a systémom vyjadrí pomocou diagramov spolupráce. Interakcie medzi aktívnymi entitami a systémom vyjadrí pomocou sekvenčných diagramov.

3. charakterizuj prostredie – analytik musí charakterizovať prostredie pomocou nasledujúcich pojmov definovaných v slovníku metodiky Adelfe – či je prostredie:

- prístupné alebo neprístupné
- deterministické alebo nedeterministické
- statické alebo dynamické
- nespojité alebo spojité

Analytik charakterizuje prostredie pomocou špecifického slovníka a preverením vstupných a výstupných rozhraní. Výstupom tejto aktivity je prvotný dokument definície prostredia.

**Aktivita 7: Stanovenie use-case-ov**: Táto aktivita má za cieľ objasniť funkcionality vyvíjaného systému (použitím use-case notácie UML) a vytvoriť model funkčného popisu, ktorý sa zahrnie do dokumentu popisu prostredia. Pozostáva z troch krokov:

1. spíš zoznam use-caseov – je potrebné spísať zoznam use-caseov vyskytujúcich sa v popisovanom systéme
2. identifikuj chyby spolupráce – cieľom je odhaliť chybné interakcie medzi entitami a systémom
3. vypracuj detailné sekvenčné diagramy – pre každý use-case identifikovaný v predchádzajúcich krokoch je potrebné vypracovať zodpovedajúci sekvenčný diagram

**Aktivita 8: Vytvorenie prototypov užívateľských rozhraní:** Analytik v dokumente „prototyp UI“ špecifikuje pomocou akých užívateľských rozhraní užívateľ interaguje so systémom, rovnako popíše aj interakcie medzi viacerými užívateľskými rozhraniami.

**Aktivita 9: Kontrola a uznanie platnosti prototypov užívateľských rozhraní**: Užívateľské rozhrania popísane v dokumente „prototyp UI“ je potrebné ohodnotiť na základe funkčných a nie-funkčných vlastností. Pri odhalení nedostatkov alebo neschválení koncovým užívateľom je potrebné dokument opätovne prepracovať v predchádzajúcej aktivite.

### 2.3.3 Disciplína analýzy

**Aktivita 10: Analýza domény a štúdium architektúry**: Analýza domény je statický pohľad a abstrakcia reálneho sveta je získaná z dokumentu požiadaviek a z dokumentu kľúčových slov. Dizajnér na základe use-caseov rozdelí systém na konkrétne a abstraktné entity. Výsledkom tejto aktivity je množina entít v predbežných diagramoch tried, ktoré sú zapísané v dokumente architektúry softvéru.

Pozostáva z troch krokov: 1. identifikuj triedy – na základe skôr definovaných use-caseov, zodpovedajúcich sekvenčných diagramov a dokumentu kľúčových slov, analytik identifikuje potrebné triedy. Počas identifikovania tried môže podľa potreby modifikovať aktívne a pasívne entity zistené počas charakterizovania prostredia.

2. analyzuj vzťahy medzi triedami – analytik analyzuje vzťahy medzi identifikovanými triedami študovaním use-caseov a sekvenčných diagramov.

3. zostroj počiatočné diagramy tried – po identifikovaní tried a vzťahov medzi nimi nasleduje vytvorenie počiatočných diagramov tried, ktoré sa uložia do dokumentu architektúry softvéru.

**Aktivita 11: Opodstatnenosť použitia adaptívneho multi-agentového systému:** Analytik rozhodne o nutnosti použitia technológie AMAS pre vyvíjaný systém. Musí sa rozhodnúť aj podľa lokálneho aj podľa globálneho hľadiska. Z globálneho pohľadu musí zodpovedať na otázku, či je adaptívny multi-agentový systém potrebný na implementáciu hľadaného systému. Z lokálneho pohľadu zase na otázku, či niektoré agenty je potrebné implementovať ako adaptívne multi-agentové systémy, t.j. či nie je potrebná ďalšia dekompozícia vyvíjaného systému.

**Aktivita 12: Identifikácia agentov:** Analytik rozhodne, či niektoré skôr identifikované entity a triedy môžu byť reprezentované v systéme ako spolupracujúce agenty, a takto vylepší dokument architektúry softvéru. Prvým krokom je pre každú entitu identifikovanú v aktivite charakteristika prostredia rozhodnúť, či:

- je autonómna
- sa snaží dosiahnuť lokálny cieľ
- musí spolupracovať s inými entitami – ak áno, či má čiastočné vedomosti o prostredí a či má schopnosti rokovať a vyjednávať s inými entitami

Entity spĺňajúce tieto kritériá môžeme prehlásiť za agentov.

Druhým krokom je identifikovanie potenciálne spolupracujúcich entít. Počas interagovania entity s inými entitami alebo s prostredím, môže naraziť na problém pri dodržiavaní protokolu alebo pri obsahovom nedorozumení. Tieto zlyhania sú dôsledkom použitia dynamického prostredia alebo otvoreného systému a nazývajú sa nespolupracujúce situácie. Pre každú entitu identifikovanú v predchádzajúcom kroku je potrebné rozhodnúť, či sa má premiestňovať v dynamickom prostredí, či má čeliť chybám spolupráce, a či má ošetrovať nespolupracujúce situácie. Posledným krokom tejto aktivity je rozpoznanie agentov. Entity spĺňajúce niektorú z podmienok z predchádzajúceho kroku môžeme prehlásiť za agentov a výsledok zapísať do dokumentu architektúry softvéru.

**Aktivita 13: Štúdium interakcií medzi rôznymi entitami:** Analytik logicky uvažuje nad vzťahmi medzi aktívnymi a pasívnymi entitami, medzi aktívnymi entitami, a medzi agentmi. Sekvenčné diagramy, diagramy spolupráce a protokolové diagramy dávajú konečnú formu dokumentu architektúry softvéru a dokumentu popisu prostredia.

### 2.3.4 Disciplína dizajnu

**Aktivita 14: Štúdium detailnej architektúry a agentového modelu:** Dizajnér objektov definuje vo vyvíjanom systéme rozdielne komponenty a hľadá možnosť opakovaného použitia už existujúcich komponentov za účelom navrhnutia architektúry systému a vytvorením dokumentu detailnej architektúry. Pozostáva zo štyroch krokov:

1. určenie balíkov
2. určenie tried – zahŕňa rozdelenie počiatočných diagramov tried získaných v aktivite „analýza štruktúry a zistenie architektúry“ do balíkov určených v predchádzajúcom kroku
3. použitie dizajnových vzorov – dizajnér zistí, či predtým definovaná architektúra nemôže byť vylepšená použitím nejakých dizajnových vzorov a opakovane použiteľných komponentov
4. detailné vypracovanie diagramov tried – cieľom tohto kroku je uzavrieť predchádzajúce kroky pospájaním navrhnutých komponentov a balíkov v UML diagramoch a vytvoriť počiatočný dokument detailnej architektúry

**Aktivita 15: Štúdium jazykov interakcií:** Agenty sa vzájomne ovplyvňujú komunikovaním a pre každý scenár definovaný v aktivitách 7 a 13 musí dizajnér agentov popísať výmeny informácií medzi agentmi. Na popis interakčných protokolov použije diagramy protokolov z AUML notácie. Výsledkom je prvotný dokument jazykov interakcií. Definovanie jazykov nie je potrebné ak agenty nekomunikujú priamo, ale len nepriamo - pomocou prostredia.

**Aktivita 16: Návrh agentov:** Dizajnér agentov skúma životný cyklus agentov pozostávajúci z vnímania, rozhodovania a konania pomocou nasledujúcich krokov:

1. definuj zručnosti – zručnosti sú vedomosti agenta o doméne, ktoré mu umožňujú konať. Definovanie prebieha na úrovni metód a/alebo atribútov.
2. definuj talent a nadanie – agent týmito schopnosťami vyvodzuje závery o doméne alebo o jeho vnímaní prostredia

3. urči jazyky interakcií – dizajnér vyberie z množiny protokolov definovaných v aktivite 15 protokol používaný agentami
4. urči znázornenie sveta – medzi metódy a/alebo atribúty sú pridané také, ktoré popisujú vnímanie seba, sveta, prostredia a ostatných agentov
5. definuj model nekooperatívnej situácie - dizajnér agentov špecifikuje pre každého identifikovaného agenta jeho nekooperatívne situácie a postup ako sa z nich dostať

**Aktivity 17,18: Rýchle prototypovanie a zlepšenie dizajnových modelov:** Dizajnér agentov testuje správanie agentov a modifikuje komponenty agentov pomocou návratu k predchádzajúcej aktivite. Taktiež kompletizuje predchádzajúce špecifikácie modelov a pomocou stavových diagramov navrhuje dynamické správanie entít vyskytujúcich sa v systéme.

## 2.4 Metodika TROPOS

Metodika TROPOS[16][17][18][19] je agentovo-orientovaná metodika vývoja softvéru. Medzi jej základné charakteristiky patrí definovanie úrovne poznania pojmov – t.j. ktoré pojmy sú použité v akých disciplínach vývoja, napr. pojmy agent, cieľ a plán sú použité vo všetkých disciplínach. Rozhodujúcu úlohu zohráva disciplína špecifikovania požiadaviek, keď sa analyzuje vyvíjaný systém a prostredie.

Metodika Tropos je založená na idei budovania konceptuálneho modelu, ktorý je inkrementálne zjemňovaný a rozširovaný z modelu skorých požiadaviek a prebieha v piatich po sebe nasledujúcich disciplínach: disciplína analýzy skorých požiadaviek, analýzy neskorých požiadaviek, architektonického dizajnu, detailného dizajnu a implementácie.

### 2.4.1 Disciplína analýza skorých požiadaviek

V disciplíne analýza skorých požiadaviek sa metodika Tropos zameriava na pochopenie problému domény pomocou štúdia existujúceho organizačného nastavenia a koordinačných procesov, ktoré charakterizujú správanie elementov domény. Sociálni aktori a softvérové systémy, ktoré sú už reprezentované v doméne, sú modelované s ich individuálnymi cieľmi. V tejto disciplíne sú identifikovaní relevantní stakeholderi a sú reprezentovaní ako aktori, pričom ich zámery sú reprezentované ako ciele.

### 2.4.2 Disciplína analýza neskorých požiadaviek

V disciplíne analýza neskorých požiadaviek sa metodika Tropos zaoberá hlavne vyvíjaným systémom, ktorý je v modeli reprezentovaný ako nový aktor, ktorý prevezme zodpovednosť za špecifické ciele sociálnych agentov. Medzi týmto novým aktorom a existujúcimi sociálnymi aktormi sa zisťujú a navrhujú nové závislosti, a taktiež môžu byť zmenené existujúce závislosti medzi sociálnymi aktormi.

### 2.4.3 Disciplína architektonický dizajn

Disciplína architektonický dizajn definuje globálnu architektúru systému pomocou podsystémov reprezentovaných ako aktori. K týmto aktorom sú pridelené podciele a podúlohy cieľov a úloh pridelených systému. Koordináciu procesu medzi komponentmi systému popisujú závislosti medzi podaktormi. Identifikovaním podaktorov systému je možné ďalej zjemňovať závislosti medzi používateľom a systémom, ktoré boli špecifikované počas disciplíny analýzy neskorých požiadaviek. Každý aktor je charakteristický množinou sociálnych/spoločenských schopností, ktoré poskytujú koordinačné mechanizmy požadované koordinačnými procesmi špecifikovanými závislosťami medzi aktormi. Výsledkom architektonického dizajnu je mapovanie podaktorov systému do množiny agentov.

### 2.4.4 Disciplína detailný dizajn

Detailný dizajn pomáha špecifikovať agenta na najnižšej úrovni. V tejto disciplíne je už obvykle zvolená implementačná platforma, čo uľahčuje priame mapovanie detailného dizajnu do kódu. Napríklad vzhľadom na platformu BDI je schopnosť každého agenta definovaná v pojmoch presvedčenia, plánov a udalostí, pričom každý plán je definovaný atomickými akciami. Systémoví aktori sú definovaní podrobnejšie, špecifikujú sa aj komunikačné a koordinačné protokoly.

## 2.5 Multiagent Systems Engineering Methodology (MaSE)

Táto metodika pokrýva kompletný softvérový životný cyklus multi-agentových systémov. MaSE[20][21][22] nešpecifikuje podrobne architektúru systému, agentovú architektúru, konkrétny programovací jazyk alebo komunikačný protokol – tieto

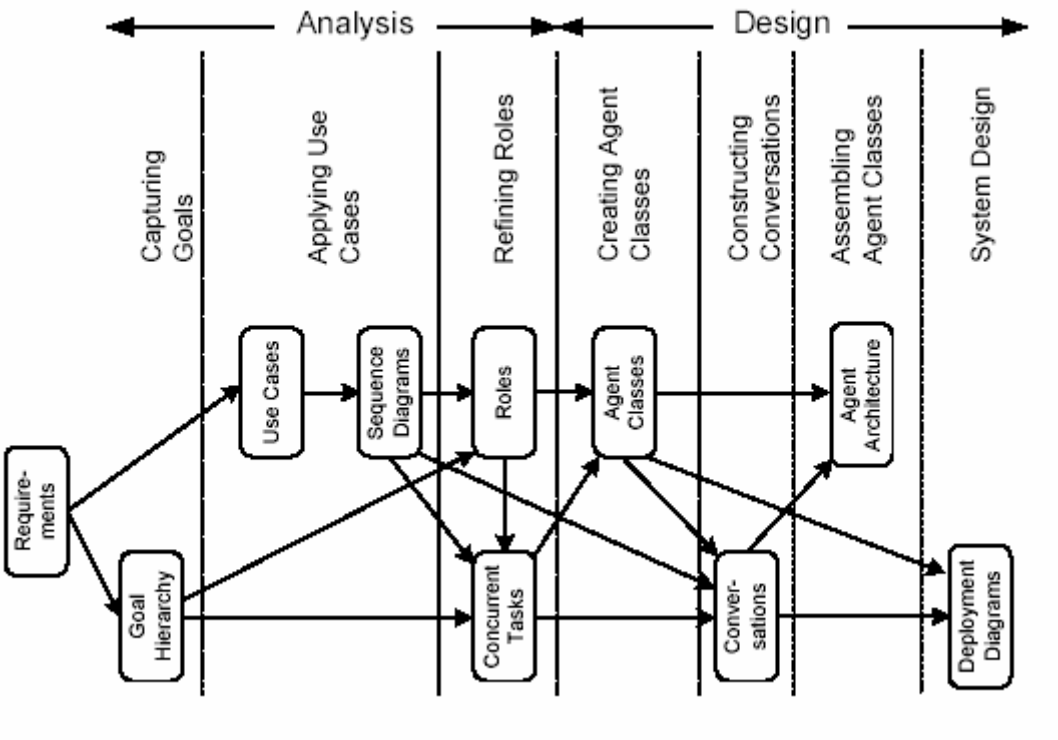
voľby sú prenechané na dizajnéra systému. Disciplíny analýzy a dizajnu sú pokryté siedmymi aktivitami, ktoré vytvárajú celkovo deväť modelov. Prvou aktivitou je zhromažďovanie cieľov. Vstupom prvej aktivity je špecifikácia systémových požiadaviek. Táto aktivita sa zaoberá transformovaním špecifikácie systému na ciele, ktoré je možno zdokumentovať do hierarchického diagramu cieľov. Hierarchický diagram cieľov je vizuálny prostriedok znázornenia cieľov v hierarchickom formáte, pričom všetky ciele na rovnakej horizontálnej úrovni patria do rovnakej oblasti funkcionality a ich podciele ich môžu svojou funkcionalitou premiestniť do vyššej úrovne.

Druhá aktivita pozostáva z vytvorenia use-case-ov z pôvodných špecifikácií systému pomocou použitia štandardných objektovo-orientovaných metód. Pre každý use-case je vytvorený aspoň jeden sekvenčný diagram. Sekvenčné diagramy modelujú posielanie správ medzi triedami objektov alebo medzi rolami agentov. Treťou aktivitou je transformácia cieľov na role. Wood a DeLoach (2000) navrhujú, že väčšina cieľov najnižšej úrovne z hierarchického diagramu cieľov sa môže stať rolami, i keď priznávajú, že v prípade veľkej podobnosti medzi dvomi cieľmi je možné skombinovať tieto ciele do jednej role. Taktiež uvádzajú, že je potrebné každému rozhraniu s externým zdrojom alebo s mobilným agentom priradiť rolu.

Role sú zdokumentované v modeli rolí. Po definovaní rolí nasleduje vytvorenie agentových tried, ktoré sú kombináciou rolí a konverzácií. Jednej agentovej roli štandardne zodpovedá jedna agentová trieda, avšak dizajnér môže občas skombinovať viacero rolí do jednej agentovej triedy, alebo rozdeliť rolu do viacerých tried. K skladaniu rolí dochádza vtedy, keď dve agentové role často alebo intenzívne (ak je počas konverzácie prenášané veľké množstvo údajov) komunikujú. Výsledkom tejto štvrtej aktivity je vytvorený diagram agentových tried zobrazujúci komunikácie medzi jednotlivými agentovými triedami. Piatou aktivitou je definovanie koordinačného protokolu medzi dvomi agentmi. Pre každú konverzáciu sú vytvorené dva komunikačné diagramy tried – jeden pre iniciátora a druhý pre toho kto odpovedá, pričom každý z nich je reprezentovaný ako konečno-stavový automat.

Kompletizácia agentov zahŕňa definovanie vnútornej architektúry predtým definovaných agentových tried. Odporúčaných je päť architektonických vzorov: viera-zámer-odhodlanie (BDI), reaktívny, plánovací, vedomostný, a užívateľom definovaná architektúra. Každý architektonický vzor navrhuje množinu komponentov k nemu asociovaných. Dizajnér sa môže rozhodnúť použiť existujúce komponenty,

alebo definovať svoje vlastné, pričom každý komponent môže byť rozdelený na menšie komponenty. Ak sú volajúce a volané komponenty uložené v rovnakom agentovi, tak sú komponenty spojené pomocou vnútorných agentových konektorov. Pri spojení komponentov pomocou vonkajších agentových konektorov sú volajúce a volané komponenty uložené v rozdielnych agentoch. Posledná aktivita sa zaoberá konkretizovaním agentových tried. Na reprezentovanie počtu inštancií agentov, typov agentov a ich umiestnenie v systéme, je vytvorený diagram rozmiestnenia. Podobne ako diagram agentových tried aj diagram rozmiestnenia zahŕňa inštancie agentov a ich konverzácie.




*Obr. č.19: Disciplíny metodiky MASE.*

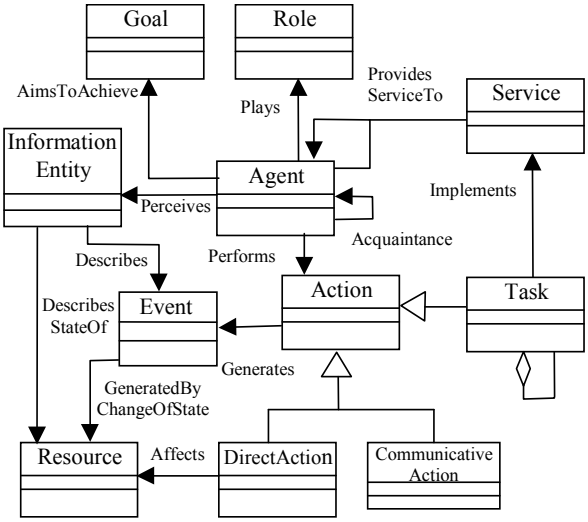
## 2.6 MESSAGE

Metodika MESSAGE[23][24][25][26] (Methodology for Engineering systems of software agents) sa zaoberá agentovými systémami, poskytuje im modely pre disciplíny analýzy a dizajnu a popisuje ako tieto modely vytvoriť. Ďalšie disciplíny – disciplína implementácie, testovania a nasadenia neboli z dôvodu závislosti od použitia konkrétnej architektúry navrhnuté.

Vytvorenie metodík AOSE sa delí na dva základné smery:

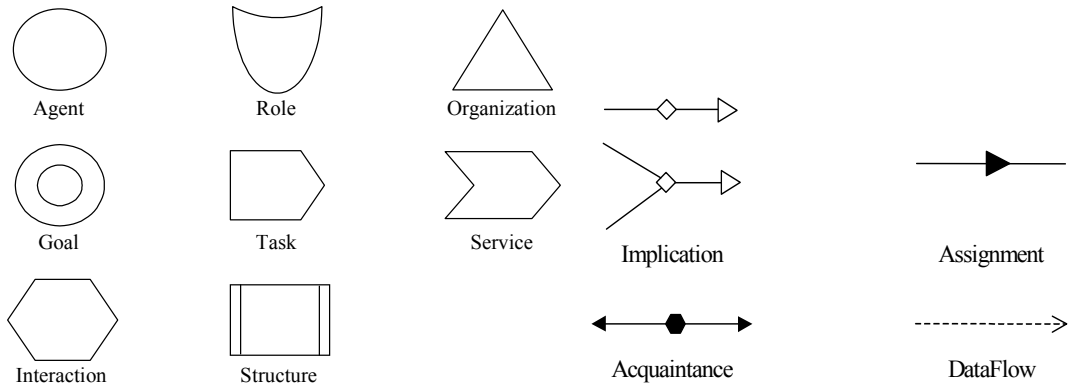
Prvý smer vychádza z existujúcej metodiky softvérového inžinierstva – t.j. rozšírením UML o agentové prvky na A-UML. Táto notácia je použiteľná a bola použitá aj v Message, ale nezachytáva podstatu agenta v jeho centre, t.j. špecifikovanie správania objektu v pojmoch interakčného protokolu nie je zhodné s agentom.

Druhý smer vychádza z agentovej teórie, s hlavným zameraním sa na disciplíny analýzy a dizajnu. Message vychádza z modelov analýzy a dizajnu z metodológie GAIA[06] a MAS-CommonKADS[27]. GAIA má dva modely analýzy a tri modely dizajnu. Napriek tomu, že modely analýzy sú založené na dobre definovaných konceptoch, reprezentujú len podmnožinu konceptov požadovaných pre agentovo-orientovanú analýzu. Dizajnové modely nie sú jasne popísané, a autori na popis detailnejšieho dizajnu uvažujú o použití OO metódy. MAS-CommonKADS má šesť modelov pre analýzu a tri pre dizajn. Aj keď sú tieto modely súhrnné, táto metóda má nedostatok v zjednotení sémantiky frameworku a notácie. Napriek tomu vyzerajú byť ciele analýzových techník veľmi použiteľné – ich rozsah je od informačných až po formálne, a pokrývajú funkčné aj nie-funkčné ciele analýzy.




*Obr. č.20: Koncept Message [23]*

Message notácia modelovania rozširuje notáciu UML s konceptmi sústredenými na agenty (viď obrázok č.20). Za základný model iteratívneho procesu modelovania bol zobraný model RUP a Message špecifikuje procesy analýzy a dizajnu výlučne v zmysle modelovacieho jazyka Message.




*Obr. č.21: Notácia použitá v metodike Message.*

### 2.6.1 Pohľady modelu analýzy:

Metodika Message ponúka možnosť vytvorenia rôznych pohľadov na vyvíjaný systém, čo uľahčí jeho pochopenie z globálneho hľadiska.

**Organizačný pohľad** zobrazuje konkrétne entity (agentov, organizácie, úlohy, zdroje) v systéme, ich správanie a niektoré (najdôležitejšie) vzťahy medzi nimi.

**Cieľov-úlohový pohľad** zobrazuje závislosti medzi cieľmi, úlohami a stavmi, môžu byť pospájané logickými závislosťami medzi nimi.

**Agentovo-úlohový pohľad** používa pre každého agenta a jeho rolu sa schému na zobrazenie jeho charakteristík.

**Interakčný pohľad** zobrazuje iniciátora, spolupracovníkov a motivátora pre každú interakciu medzi agentami a rolami – nimi dodané relevantné informácie.

**Doménový pohľad** zobrazuje doménovo špecifické koncepty a vzťahy relevantné pre systém pri vývoji – môže byť reprezentovaný diagramami tried v UML, pričom triedy reprezentujú doménovo špecifické koncepty a pomenované asociácie reprezentujú doménovo špecifické vzťahy.

### 2.6.2 Proces modelovania:

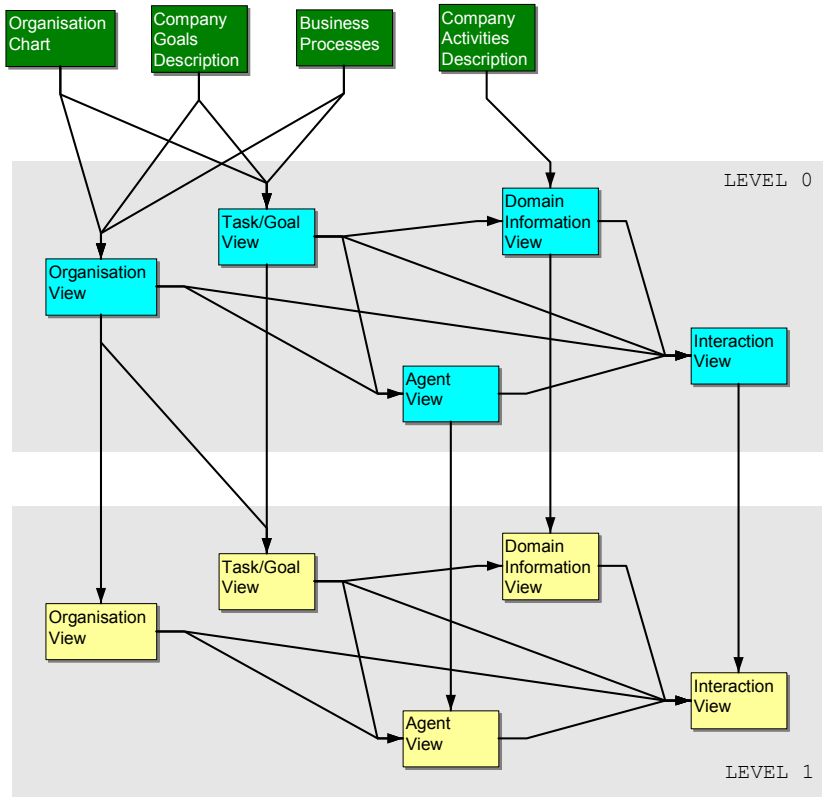
Dôvodom analýzy je vytvoriť špecifikáciu = model systému (kolekciu pohľadov), ktorý bude vyvíjaný a jeho prostredie. Model analýzy je vytváraný postupným zjemňovaním a vylepšovaním.

Najvyššia úroveň dekompozície sa označuje ako úroveň 0. Táto úroveň sa zaoberá definovaním systému, ktorý bude vyvíjaný, s rešpektovaním stakeholderov a prostredia. Systém je chápaný ako množina organizácií, ktoré reagujú so zdrojmi, aktormi alebo inými organizáciami. Aktormi môžu byť ľudia (užívatelia) alebo iné

existujúce agenty. Následným zjemňovaním výsledkov vytvárania modelov dostávame úroveň 1, úroveň 2, atď.

V úrovni 0 začína proces modelovania vytvárať pohľad organizačný a cieľovo/úlohový. Tieto pohľady môžu pôsobiť ako vstupy na vytvorenie pohľadov agentovo/úlohového a doménového pohľadu. A nakoniec je z údajov týchto modelov vytvorený interakčný pohľad. Model úrovne 0 dáva celkový pohľad na systém, jeho prostredie a jeho globálnu funkcionalitu. Zjemňovanie úrovne 0 sa zameriava na identifikovanie entít a ich vzťahov podľa meta-modelu. Ďalšie informácie o vnútornej štruktúre a správaní týchto entít sú postupne vkladané do ďalších úrovní.

V úrovni 1 je štruktúra a správanie entít (ako napr. organizácií, agentov, úloh, cieľov doménových entít) definované ďalšími úrovňami, ktoré môžu byť potrebné na analyzovanie špecifických aspektov systému spolu s funkčnými a nefunkčnými požiadavkami, ako sú napr. výkon, rozdelenie, odolnosť voči chybám a bezpečnosť. Na ďalších nižších úrovniach musí byť zabezpečená zhoda s vyššími úrovňami.




*Obr. č.22: Všeobecný workflow procesu analýzy Message [23]*

**Stratégie analýzového zjemňovania** (existuje niekoľko stratégií na zjemnenie modelov úrovne 0):

**Organizačno-centrovaný prístup** sa zameriava na analýzu celkových vlastností – ako aj štruktúry systému, tak aj poskytovaných služieb, celkových úloh a cieľov, hlavných úloh a zdrojov. Po odhalení agentov môže byť analyzovaná ich kooperácia, konflikty a riešenie konfliktov.

**Agentovo-zameraný prístup** sa zameriava na identifikovanie agentov potrebných na zabezpečenie funkčnosti systému. Najvhodnejšia organizácia je identifikovaná podľa systémových požiadaviek.

**Interakčno-orientovaný prístup** navrhuje postupné zjemňovanie scenárov, ktoré charakterizuje vnútorné a vonkajšie správanie organizácií a agentov. Tieto scenáre sú zdrojom pre charakterizovanie úloh, cieľov, správ, protokolov a doménových entít.

**Cieľovo-úlohové dekompozičné prístupy** sú založené na funkčnej dekompozícii. Role systému, ciele a úlohy sú systematicky analyzované za účelom zistenia podmienok rozlíšenia, metód riešenia problémov, dekompozície a opravovania chýb. Predpoklady úlohy, štruktúry úlohy, výstupy úlohy a následné podmienky úlohy môžu rozoznať, aké doménové entity sú potrebné. Agenty, ktoré hrajú hlavné role, musia vykonávať ciele a úlohy. Takže celkovým pohľadom na štruktúru cieľov a úloh z cieľovo-úlohového pohľadu môžu byť rozhodnutia vytvorené na najvhodnejších agentoch a organizačnej štruktúre na dosiahnutie daných cieľov a úloh.

Rozlišujeme dva základné kroky modelovania:

1. Prvým krokom je identifikovanie entít a špecifikovanie, ako súvisia s ostatnými entitami. Tento krok sa odporúča vykonať v troch paralelných disciplínach:
   a) identifikovanie konkrétnych entít a príbuzenských a organizačných vzťahov medzi nimi – treba pracovať v rámci organizačného pohľadu
   b) identifikovanie cieľov a úloh, logických a dočasných vzťahov medzi nimi– treba pracovať v rámci pohľadu cieľov/úloh
   c) identifikovanie informácií a iných doménových entít a vzťahov medzi nimi– treba pracovať v rámci pohľadu cieľov/úloh

2. Druhým krokom je popisovanie vnútorných detailov a entít, hlavne agentov, rolí, organizácií a interakcií – treba pracovať v rámci pohľadu agent/úloha a interakčného pohľadu. Môže to tiež vyvolať spracovanie pohľadov cieľov/úloh a doménového pohľadu.

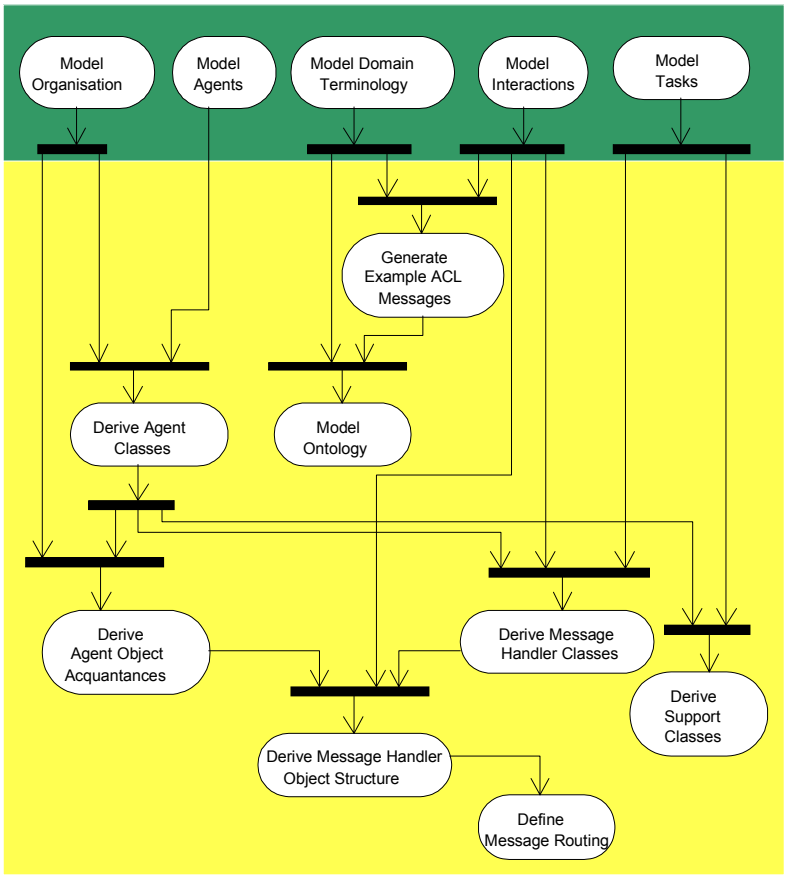
Model bude obsahovať minimálne dve úrovne abstrakcie: Úroveň 0 obsahuje definíciu vyvíjaného systému (reprezentovaného ako jednoduchá entita) s rešpektovaním jeho prostredia a stakeholderov. Úroveň 1 zjemňuje vyvíjaný systém do organizácie interagujúcich agentov. Ďalšie úrovne zjemňovania sa dajú získať rekurzívnym procesom obsahujúcim úroveň 0 a 1.

### 2.6.3 Disciplína dizajnu:

Metodika Message uvažuje o dvoch metódach dizajnu:

1. vychádza z agentovej architektúry a MAS – vychádza z toho, že agent je považovaný za entitu, ktorá je viac ako trieda – agent je chápaný ako podsystém s vnútornou architektúrou, ktorá definuje vzťahy rôznych agentových komponentov. Tieto komponenty sú výpočtové entity, ktoré sú identifikované a vytvárané transformáciou a zjemňovaním z modelov disciplíny analýzy. Správanie niektorého z týchto komponentov môže byť komplexné (napr. riadenie agenta), alebo jednoduché (reaktívne agenty). Dizajnový proces vychádza z organizačného modelu za účelom pridelenia zodpovedností, definovania vzťahov medzi agentmi a modelovania sociálneho správania.

2. agentovo-platformovo orientovaný = predpokladá, že každý agent môže byť mapovaný na triedu. Toto je hlavne prevzaté z aplikácií agentových modelov, ktoré sú podporované nástrojmi na tvorbu agentov – ako napr. Jade alebo FIPA-OS, v ktorých je jedna trieda Agent, z ktorej sú odvodené špecifické typy agentov. Tento prístup je platný, keď agenty majú jednoduché správanie a môže byť teda modelované klasickou metódou stavového automatu – ako napr. v typicky OO jazyku Java. Hlavným problémom je ako zorganizovať interakcie medzi agentmi.



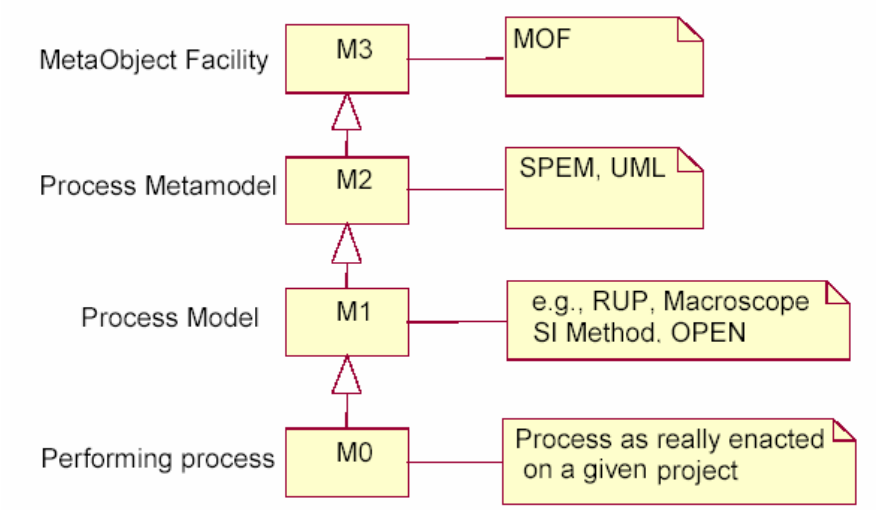

*Obr. č.23: Model produktov a definícií činností metodiky Message*

# 3 Porovnanie metodík podľa disciplín vývoja softvéru.

V tejto kapitole preskúmame prístup jednotlivých metodík k rozdeľovaniu procesu vývoja softvéru na disciplíny, pracovné postupy a aktivity. Každá zo skúmaných metodík pristupuje svojim spôsobom k pomenovávaniu disciplín a k reprezentovaniu entít a vzájomných vzťahov medzi entitami. Bolo by teda takmer nemožné hľadať spoločné a odlišné črty bez počiatočného sformalizovania, t.j. reprezentovania metodík jednotným jazykom. Jednotlivé metodiky sú taktiež reprezentované rôznym stupňom granularity. Vzniká teda potreba uviesť skúmané metodiky na spoločnú úroveň. Na sformalizovanie nám poslúži Software Process Engineering Metamodel (SPEM) [09][28]. Vyjadrením procesov vývoja pomocou všeobecného meta-modelu SPEM ľahšie odhalíme spoločné a odlišné črty jednotlivých disciplín a aktivít skúmaných metodík vývoja multi-agentových systémov.

## 3.1 Software Process Engineering Metamodel (SPEM)

SPEM je meta-model na definovanie procesov a ich komponentov. Používa sa na popis konkrétneho procesu vývoja softvéru alebo skupiny súvisiacich procesov vývoja softvéru. Vychádza z objektovo-orientovaného prístupu a na popis používa UML[29]. Slúži ako vzor pre popis procesu opisujúceho reálny produkčný proces. OMG definovala štvorvrstvovú architektúru modelovania(zovšeobecňovania) procesov. Na úrovni M0 je skutočný produkčný proces. Definícia modelu jemu zodpovedajúcemu procesu je na úrovni M1. Pre softvérové inžinierstvo je takýmto modelom Rational Unified Process. Rovnako sa v tejto úrovni nachádzajú aj použitia modelu RUP na konkrétny projekt. SPEM je v hierarchii na úrovni M2 a jeho cieľom je slúžiť ako všeobecný vzor pre modely z úrovni M1. Špecifikácia SPEM je štruktúrovaná ako UML profil a poskytuje tiež kompletný metamodel založený na MOF. Takáto špecifikácia napomáha vzájomnému previazaniu a použitiu UML nástrojov a MOF nástrojov. Z dôvodu všeobecného použitia na popis procesov vývoja softvéru je SPEM navrhnutý s minimálnou množinou modelovacích elementov. Bolo by pravdepodobne neefektívne snažiť sa vytvoriť komplexnejší a detailnejší model. Jeho snahou je vyhovieť požiadavkám väčšine existujúcich a popísaných procesov vývoja softvéru. Keďže takmer každý proces používa trochu odlišnú terminológiu, resp. rovnaké termíny s čiastočne odlišným významom, tak bolo potrebné zadefinovať jednotnú terminológiu. Na vypracovaní jednotnej schválenej terminológie pracuje organizácia FIPA.




*Obr. č.24: Architektúra modelovania procesov [09]*

## 3.2 Disciplína zberu požiadaviek

### 3.1.1 Message, Gaia, Adelfe

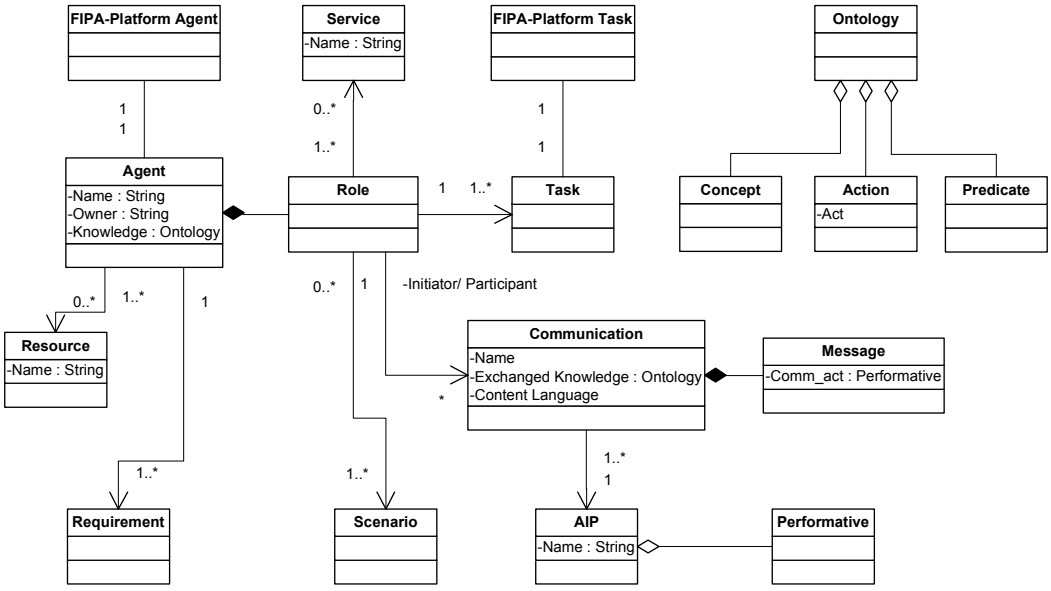
Na úvod si stručne prejdeme jednotlivé metodiky z hľadiska disciplíny zberu požiadaviek. Prvým extrémom sú metodiky, ktoré nevravia priamo o disciplíne zberu požiadaviek. Prvou takou je metodika MESSAGE, ktorej disciplína analýzy obsahuje rôzne úrovne dekompozície, pričom dekompozíciu úrovne 0 možno považovať za disciplínu zberu požiadaviek. Ďalšou takou je metodika Gaia, ktorá sa napríklad nezaoberá popisom spôsobu, akým má systémový analytik identifikovať a získať požiadavky, ale už pri vstupe do nasledujúcej disciplíny (disciplíny analýzy) predpokladá existenciu dokumentu so špecifikovanými požiadavkami.

Druhým extrémom sú metodiky Tropos a Adelfe, ktoré majú až dve disciplíny zberu požiadaviek – disciplínu skorých a disciplínu neskorých požiadaviek.

### 3.1.2 PASSI a MaSE

Metodika PASSI pomenúva prvú disciplínu disciplínou systémových požiadaviek a zahŕňa v nej aj aktivity patriace skôr do disciplíny analýzy. Metodika MaSE, ktorá nepoužíva názvoslovie známe z RUP (Rational Unified Process), ale pomenúva disciplíny podľa aktivít prebiehajúcich v jednotlivých disciplínach.

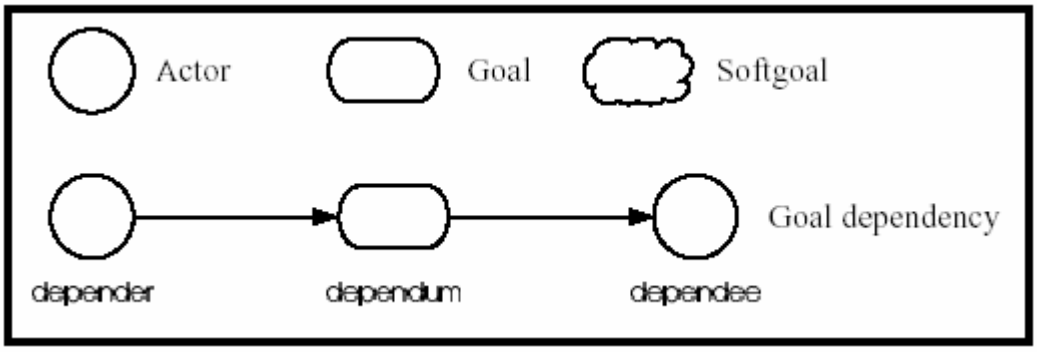
Ako vidieť, tvorcovia metodík pristupovali k rozdeľovaniu procesu vývoja rôzne, a my sa ho teda budeme snažiť zovšeobecniť. Prejdime už jednotlivým metodikám.




*Obr. č.25: MAS meta-model zvolený v PASSI [10]*

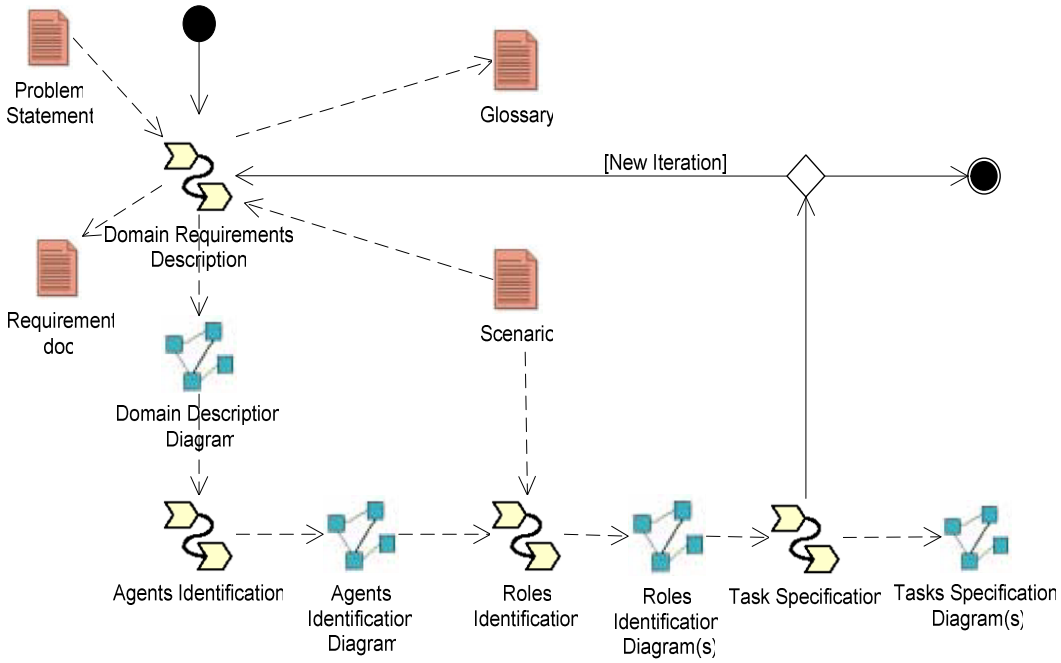
### 3.2.1 TROPOS

Tropos sa v disciplíne skorých požiadaviek zaoberá porozumením problémov domény, identifikovaním jej stakeholderov, štúdiom organizačnej situácie a koordinácie procesov. Výsledkom je model skorých požiadaviek, ktorý obsahuje závislosti medzi aktormi a cieľmi, ako je načrtnuté na obrázku č.26. Základnou ideou Troposu je, že koordináciu je možné modelovať pomocou závislostí medzi aktormi a následným zjemňovaním počiatočného modelu sú vytvárané modely neskorších disciplín procesu vývoja. Počiatočný model je postupne dopĺňaný a rozširovaný o nové závislosti.




*Obr. č.26: Spôsob vyjadrenia závislostí medzi aktormi v metodike Tropos*

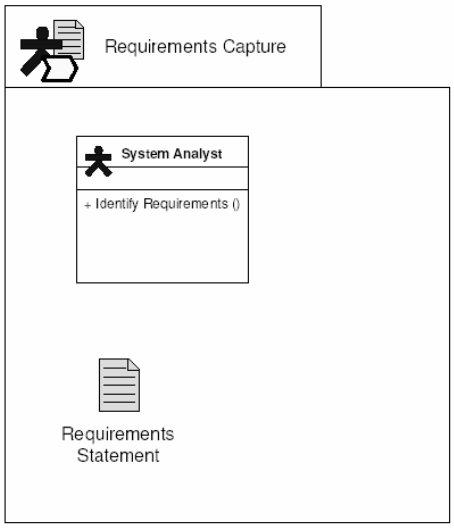
Adelfe a Passi sa príliš nelíšia v disciplíne zberu požiadaviek, i keď Adelfe v disciplíne skorých a neskorých požiadaviek pokrýva menej aktivít ako metodika Passi, ktorá už prvej disciplíne obsahuje aktivity patriace skôr do disciplíny analýzy. Ich spoločnou prvou aktivitou je odhaľovanie užívateľových požiadaviek a následný vznik dokumentov, ktorými sú slovník pojmov, scenáre a dokument požiadaviek.




*Obr. č. 27: Metodika PASSI – disciplína zberu požiadaviek:*

### 3.2.2 Gaia

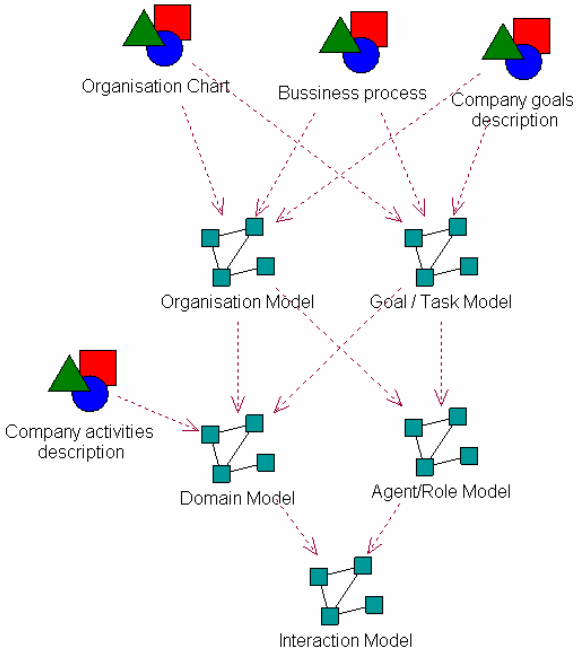
Ďalšou takou je metodika Gaia, ktorá sa napríklad nezaoberá popisom spôsobu, akým má systémový analytik identifikovať a získať požiadavky, ale už pri vstupe do nasledujúcej disciplíny (disciplíny analýzy) predpokladá existenciu dokumentu so špecifikovanými požiadavkami. Bezstarostnosť tohto prístupu vidieť na obrázku č.28.




*Obr. č. 28: Metodika Gaia - disciplína zhromažďovanie požiadaviek*

### 3.2.3 MESSAGE

MESSAGE je metodika, ktorej prvou disciplínou je disciplína analýzy. Mohlo by sa zdať, že tak ako Gaia, aj Message predpokladá už špecifikované požiadavky pri vstupe do disciplíny analýzy. Skutočnosť je však iná. Message totiž rozoznáva niekoľko úrovní disciplíny analýzy. Najvyššou úrovňou dekompozície je úroveň 0, v ktorej sú vytvárané prvotné modely. Postupným inkrementálnym zjemňovaním vytvorených modelov vzniká úroveň 1 a v prípade potreby aj ďalšie (2,...). Úroveň 0 môžeme na základe činností, ktorými sa zaoberá, zaradiť do disciplíny zberu požiadaviek. Medzi tieto činnosti patrí identifikovanie entít a vzťahov medzi nimi v prostredí vytváraného systému a vytvorenie organizačného a cieľovo-úlohového modelu.




*Obr. č. 29: Metodika Message, diagram toku dát*

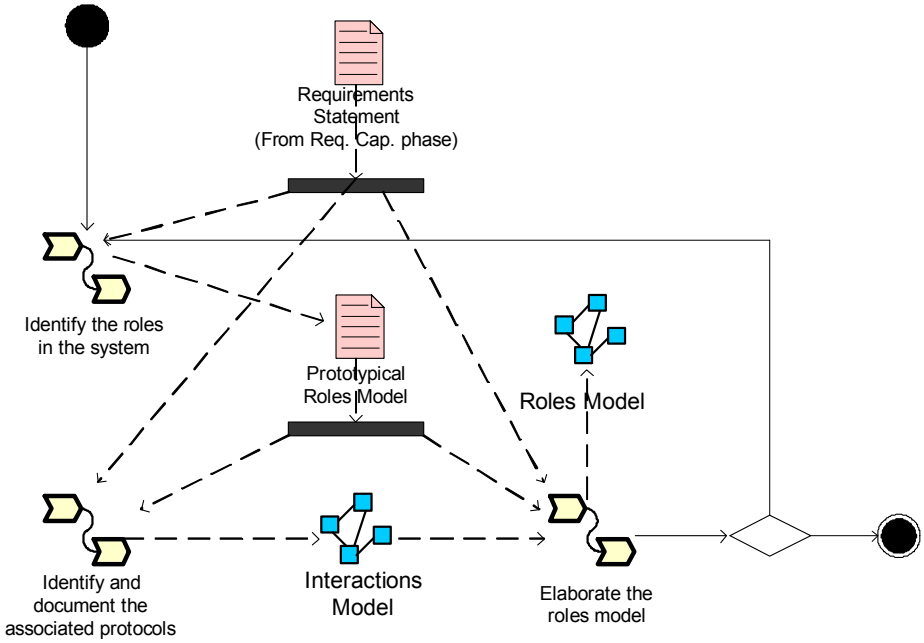
### 3.2.4 MaSE

Metodika MaSE neobsahuje disciplínu zberu požiadaviek, ale v jej disciplíne analýzy možno definíciu činnosti zhromažďovanie cieľov považovať za disciplínu zberu požiadaviek, pretože jej úlohou je identifikovanie cieľov, získanie use-case-ov a štruktúry cieľov.

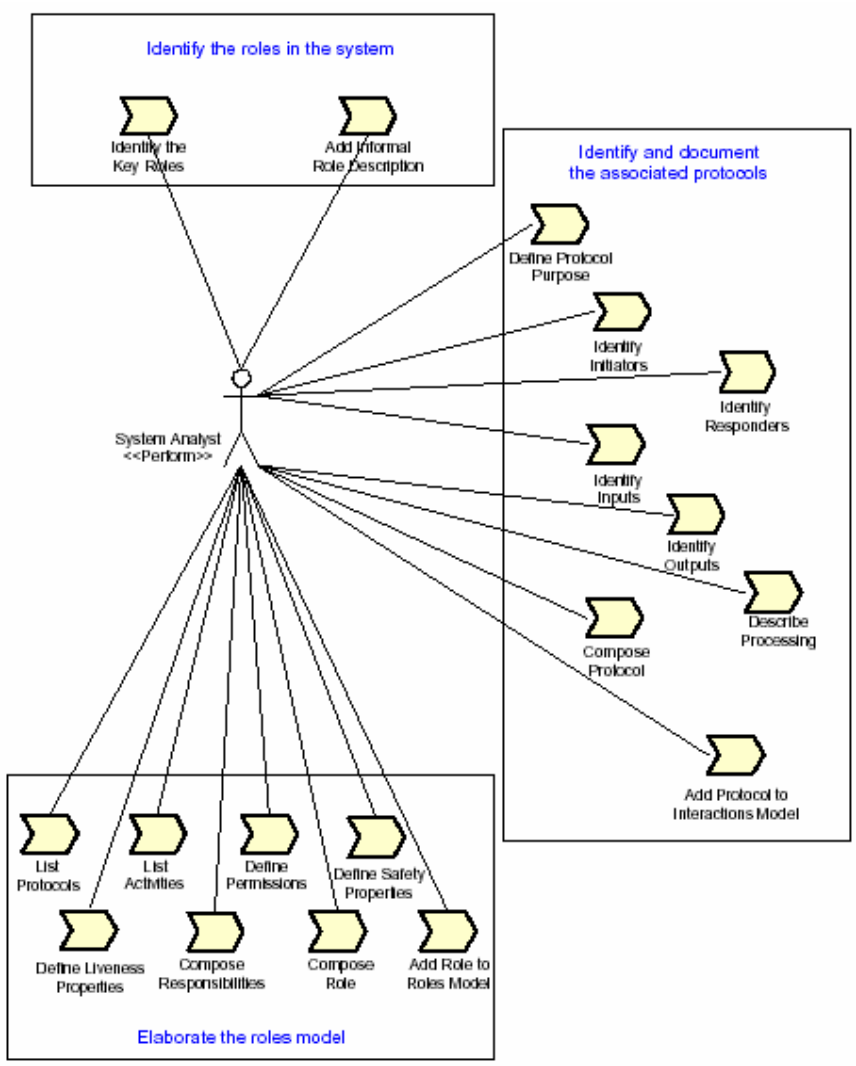
## 3.3 Disciplína analýzy

### 3.3.1 Gaia

Metodika Gaia sa v disciplíne analýzy zameriava hlavne na vytvorenie modelu rolí a modelu interakcií. Vstupom tejto disciplíny je dokument so špecifikovanými požiadavkami, pomocou ktorého sú identifikované kľúčové role v systéme a vytvorený prototyp modelu rolí.




*Obr. č. 30: Metodika Gaia, disciplína zhromažďovanie požiadaviek*



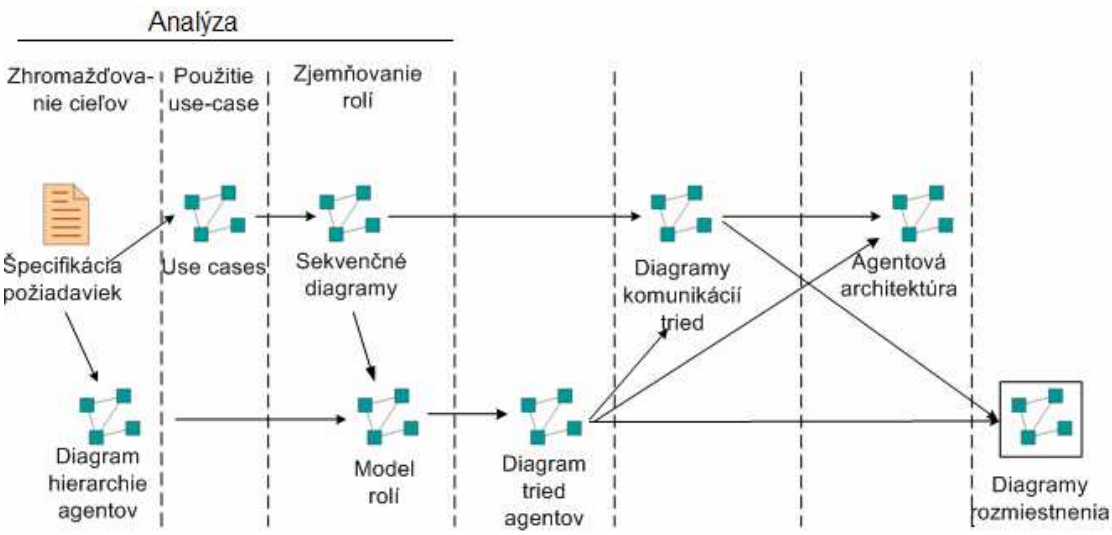

*Obr. č. 31: Metodika Gaia, diagram aktivít disciplíny zhromažďovanie požiadaviek*

### 3.3.2 PASSI

Metodika PASSI sa už v disciplíne systémových požiadaviek zaoberá vytvorením modelu identifikovaných rolí (podobne ako Gaia). Disciplínu analýzy tvoria časti disciplína systémových požiadaviek a disciplína spoločenstva agentov. Výsledkom disciplíny spoločenstva agentov sú štyri modely: model popisu ontológie domény, model popisu ontológie komunikácie, model popisu protokolov a model popisu rolí, v ktorých možno nájsť spoločné črty ako má model interakcií v metodike Gaia – konkrétne ide o identifikovanie a popis protokolov.

### 3.3.3 MASE

MaSE rozdeľuje disciplínu analýzy na tri poddisciplíny: disciplínu zhromažďovania cieľov, disciplínu použitia use-case-ov a disciplínu zjemňovania rolí. Prvú z nich sme už zaradili do disciplíny zberu požiadaviek, zvyšné dve vystihujú disciplínu analýzy. Z disciplíny zhromažďovania cieľov získame model use-case-ov a diagram hierarchie cieľov, ktoré využijeme na transformáciu cieľov na role a následný vznik modelu rolí, ako ilustruje obrázok č.32. Metodike MaSE chýba spoločná, medzi agentami zdieľaná ontológia, ktorá by vylepšila komunikáciu medzi agentmi. DiLeo (2002) [30] navrhol rozšíriť MaSE o ďalšiu disciplínu s názvom Vytvorenie ontológie, ktorá by bola medzi disciplínou použitia use-case-ov a disciplínou zjemňovania rolí a jej vstupmi by boli diagram hierarchie cieľov, model použitia use-case-ov a sekvenčný diagram.




*Obr. č. 32: Disciplíny metodiky MaSE – zvýraznená disciplína analýza*

### 3.3.4 Tropos a Message

Odlišný prístup k prechodu medzi disciplínami majú metodiky Tropos a MESSAGE. Tieto využívajú techniku cyklického rozširovania a zjemňovania prvotne vytvorených modelov.

V našom ponímaní disciplíny analýzy jej v metodike MESSAGE zodpovedá analýza úrovne 1, pretože pri prechode z úrovne 0 na 1 sa analýza zameriava na identifikovanie hlavných častí požadovanej funkcionality systému. Takto sú objavené role a typy agentov. Message umožňuje použiť jeden z viacerých prístupov

zjemňovania úrovne 0 – so zameraním na: organizáciu, agentov, interakcie, alebo ciele a úlohy.

Metodika TROPOS nepozná disciplínu analýzy, ale podľa aktivít prebiehajúcich v jej disciplíne neskorých požiadaviek a architektonického dizajnu môžeme tieto dve disciplíny do nej zaradiť. Systémové komponenty sú reprezentované ako aktori a majú pridelené podciele a podplány na základe cieľov a plánov pridelených k systému. Podľa závislostí medzi aktormi a podľa cieľov a plánov, ktoré systémoví aktori vykonávajú, sú identifikované schopnosti jednotlivých aktorov.

### 3.3.5 Adelfe

V metodike Adelfe charakterizuje disciplínu analýzy niekoľko aktivít. Prvou je analýza domény. Doménový analytik v nej vytvorí dokument architektúry softvéru, v ktorom sú identifikované entity. Následne musí agentový analytik overiť opodstatnenie použitia adaptívneho MAS. Agentový analytik pokračuje prehodnotením identifikovaných agentov v disciplíne neskorých požiadaviek, či niektoré z nich môžu byť reprezentované ako kooperatívne agenty, čo zlepší dokument architektúry softvéru. Poslednou aktivitou disciplíny analýzy je štúdium interakcií medzi entitami. Doménový analytik po skúmaní vzťahov medzi aktívnymi a pasívnymi entitami, medzi aktívnymi entitami, a medzi agentmi, vytvorí modely sekvenčných diagramov a diagramy protokolov, ktorými dá konečnú formu dokumentu architektúry softvéru a dokumentu popisu prostredia.

Model rolí je štandardným produktom disciplíny analýzy skúmaných metodík. Niektoré metodiky sa zaoberajú aj skúmaním komunikácií a vytvárajú model komunikácie.

## 3.4 Disciplína dizajnu

### 3.4.1 Adelfe

Dizajnér objektov študuje dokument architektúry softvéru a multi-agentový model so zámerom určiť rôzne komponenty (triedy, balíky, …) nachádzajúce sa v systéme, ktoré by bolo možné nahradiť už existujúcimi komponentmi, prípadne nájsť komponenty na viacnásobné použitie, pričom vytvorí prvotný dokument detailnej

architektúry. Dizajnér agentov sa usiluje popísať rôzne interakčné protokoly agentov a vytvorí z nich prvotný dokument jazykov interakcií. Následne pokračuje špecifikovaním architektúry pre každého identifikovaného agenta, t.j. špecifikuje jeho skúsenosti, schopnosti, interakčné jazyky, … , a pomocou rýchleho prototypovania testuje správanie agentov, modifikuje komponenty agentov a tak rozširuje dokument jazykov interakcií a dokument detailnej architektúry.

### 3.4.2 Gaia

Vstupmi do disciplíny dizajnu sú model rolí a model interakcií, ktoré boli vytvorené v disciplíne analýzy. Prvou úlohou dizajnéra agentov je vytvoriť agentový model, vychádzajúc z modelu rolí, v ktorom identifikuje typy agentov tvoriacich systém a inštancie agentov z týchto typov. Podľa modelu interakcií a modelu rolí, vytvorí dizajnér agentov model služieb, v ktorom sú identifikované hlavné služby potrebné na vykonanie úloh agentov. V závere tejto disciplíny ešte z agentového modelu a modelov rolí a interakcií vytvorí model okruhu známych, ktorý sa podobá na model komunikácie (známy z iných metodík), pretože sú v ňom zdokumentované linky komunikácií medzi rôznymi agentmi.

### 3.4.3 PASSI

Metodika Passi má disciplínu agentovej implementácie, ktorú podľa väčšiny aktivít a produktov v nej vytvorených, možno považovať za disciplínu dizajnu. Z výsledkov predchádzajúcich disciplín využíva disciplína agentovej implementácie model identifikácie agentov, model špecifikácie úloh a diagram ontológie komunikácie, na základe ktorých dizajnér agentov vytvorí diagramy definície štruktúry multi-agentov a diagramy správania multi-agentov.

### 3.4.4 TROPOS

Metodika TROPOS rozlišuje dve disciplíny dizajnu – disciplínu architektonického a disciplínu detailného dizajnu, pričom disciplínu architektonického dizajnu sme už zaradili v našej terminológii do disciplíny analýzy. Disciplínu dizajnu v našom ponímaní čiastočne vystihuje disciplína detailného dizajnu, ktorá sa zameriava na špecifikovanie mikro-úrovne agentov.

V tejto disciplíne sa rozhoduje aj o implementačnej platforme a vytvára sa dizajn pre požadované interakčné a komunikačné protokoly.

### 3.4.5 MESSAGE

Metodika MESSAGE uvažuje v disciplíne dizajnu o dvoch hlavných možnostiach prístupu. Prvý vychádza z organizácie multi-agentových systémov a agentovej architektúry. Agent je považovaný za niečo viac ako trieda, je chápaný ako podsystém s vnútornou architektúrou, ktorá definuje vzťahy rôznych agentových komponentov identifikovaných a rozširovaných v procese zjemňovania v disciplíne analýzy. Správanie niektorých týchto komponentov môže byť komplexné (napr. podľa schémy BDI), alebo jednoduché (napr. ako reaktívne agenty). Druhý prístup sa zameriava viac na platformu a uvažuje o tom, že každý agent môže byť mapovaný na triedu. Tento prístup je možné použiť ak správanie agentov je jednoduché a môže byť modelované klasickým OO prístupom použitím programovacích jazykov akým je napríklad Java. K tomuto prístupu je vytvorených viacero nástrojov (Jade a FIPA-OS) podporujúcich transformáciu agentových modelov mapovaním na triedy. Medzi základné aktivity, ktoré metodika MESSAGE uskutočňuje v disciplíne dizajnu, patrí detailná analýza entít, voľba agentovej architektúry, vypĺňanie zvolenej architektúry agentami, aktualizácia pohľadov existujúcich modelov a rozdeľovanie agentov a ich vzťahov podľa organizačného pohľadu do navrhnutých štruktúr.

## 3.5 Disciplíny implementácie, testovania a nasadenia

Nie sú oblasťou skúmania tejto práce. Skúmané metodiky väčšinou tiež nezastrešujú spomenuté disciplíny vývoja softvéru, nakoľko sú tieto disciplíny závislé od použitej agentovej platformy a programovacieho jazyka. Ich skúmanie môže byť podnetom ďalšieho študenta na rozšírenie tejto práce o uvedené oblasti, pričom mu môžu pomôcť poznatky obsiahnuté v diplomovej práci Norberta Zagyiho [31], kde sú už z rôznych hľadísk porovnané agentové platformy.

## 3.6 Zhrnutie

V súčasnosti už bolo pre analýzu a dizajn MAS navrhnutých niekoľko kompletných metodík [32], ale len veľmi málo z nich sa zameriava na organizačnú abstrakciu.

Metodika MASE [22] poskytuje návod na vývoj MAS založeného na viackrokových procesoch. V analýze sú použité požiadavky na definovanie use-caseov a cieľov/podcieľov, a eventuálne na identifikovanie rolí hraných agentmi a ich interakcií. V dizajne sú z výstupov disciplíny analýzy odvodené triedy agentov a interakčné protokoly, vedúce ku skompletizovaniu architektúry systému. Napriek tomu, MASE zlyhá pri identifikácii akejkoľvek abstrakcie organizácie inej ako model rolí – v disciplíne analýzy nie sú tu žiadne využitia abstrakcií, ktoré by mohli byť začlenené do organizačných pravidiel, a definícia organizačnej štruktúry je odvodená ako implicitný výstup z disciplíny analýzy. Ako je otvorene priznané tvorcami MASE, táto vlastnosť robí metodiku MASE vhodnú len pre uzavreté systémy. V súčasnosti sa autori MASE snažia rozšíriť metodiku MASE o vlastnosti potrebné pre otvorené systémy a dali jej názov O-MASE.

Metodika MESSAGE [33] využíva abstrakcie organizácie, ktoré môžu byť mapované do abstrakcií definovaných metodikou Gaia. Metodika MESSAGE definuje najmä organizáciu pomocou pojmov ako napr. *štruktúra*, rozpoznáva role hrané agentmi a ich vzájomné vzťahy, ich vzájomné interakcie. Toto sa zhoduje s konceptom štruktúry organizácie propagovanou metodikou Gaia. MESSAGE neadresuje problém identifikovania a jasného modelovania organizačných pravidiel (sú skryto zakomponované do organizačnej štruktúry) a neuznáva potrebu otvoreného dizajnu organizačnej štruktúry (čo sa ale nepriamo očakáva ako výstup disciplíny analýzy). Po ďalšie, MESSAGE je tesne prepojená s UML (a AUML), kým Gaia je všeobecnejšia a poskytuje jasné inštrukcie bez viazania sa na niektorú konkrétnu modelovaciu techniku.

Prvá verzia Gaia[34] poskytovala jasné rozdelenie medzi disciplínami analýzy a dizajnu. Napriek tomu trpela rôznymi obmedzeniami - podobnými ako MESSAGE a MASE – spôsobenými nekompletnosťou svojej množiny abstrakcií. Cieľom disciplíny analýzy metodiky Gaia je definovať kompletne do detailov vypracovaný model rolí, odvodený zo špecifikácie systému, spolu s presným opisom protokolov pre tie role, ktorých sa to týka. Toto nepriamo predpokladá, že celá organizačná štruktúra bola už vopred známa – čo nie je vždy pravda. Výrazným zameriavaním sa

na model rolí (v prvej verzii Gaia), zlyhalo identifikovanie oboch konceptov globálnych organizačných pravidiel (teda robením ich nevhodnými pre modelovanie otvorených systémov a pre riadenie správania sebeckých agentov) a modelovanie prostredia.

Nová verzia Gaia prekonáva tieto obmedzenia. Ďalšie rozšírenie Gaia bolo navrhnuté v [35], a snaží sa prekonať tieto obmedzenia (napr. reprezentovanie prostredia a organizačných pravidiel). Napriek tomu je tento návrh stále predbežný a chýba mu premyslená integrácia v rámci Gaia procesu.

Metodika TROPOS, prvýkrát navrhnutá v [36] a vylepšená v [37], zdieľa s Gaia oboje spomenuté vlastnosti - dáva dôraz na identifikáciu organizačnej štruktúry a pravidiel. Podobne ako Gaia, aj TROPOS uznal, že organizačná štruktúra je základným aspektom pre vývoj agentových systémov a že jej vhodná voľba je potrebná na splnenie funkčných aj nie-funkčných požiadaviek. TROPOS sa ale aj líši od metodiky Gaia vo viacerých aspektoch. Na jednej strane, TROPOS definuje jednotný a premyslený návod pre aktivity v disciplínach získania skorých a neskorých požiadaviek, kým disciplína analýzy metodiky Gaia sa najviac zaoberá neskorými požiadavkami. Nevylučujeme možnosť adaptovania metód a techník z cieľovo zameranej analýzy do analýzy skorých požiadaviek metodiky Gaia [38][39]. Na druhej strane metodika TROPOS nie úplne otvorene identifikuje koncept organizačných pravidiel. I keď niektoré otvorené závislosti medzi rolami sú požadované byť identifikované v disciplíne analýzy, žiadny model metodiky TROPOS nie je schopný zachytiť globálne zákony aplikovateľné na mnohé organizačné role alebo na organizáciu ako celok. Toto môže znížiť účinnosť analýzy a zvýšiť komplexnosť nasledujúcej disciplíny dizajnu.

Zatiaľ čo spomenutá metodika Gaia navrhuje radšej sekvenčný prístup vývoja softvéru, prebiehajúc priamo z disciplíny analýzy k dizajnu a potom k implementácii, vývoj softvéru sa často musí zaoberať nestálymi požiadavkami, zlými predpokladmi a chýbajúcimi informáciami. To požaduje od dizajnéra spraviť krok späť od súčasnej aktivity a možno aj opätovne premyslieť niektoré jeho skoršie rozhodnutia a závery. A v Gaia nie sú žiadne otvorené štruktúry alebo proces, ktorý by to umožňoval.

# 4 Návrh novej metodiky

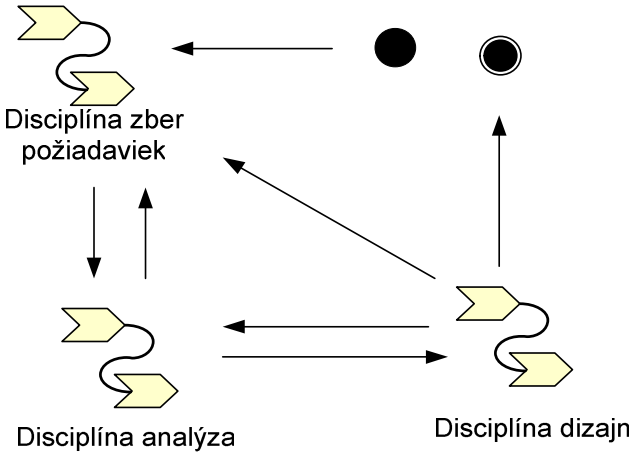
Vychádzajúc z poznatkov získaných počas štúdia metodík tvorby multi-agentových aplikácií, podrobnejšie skúmaných v kapitole 3, sa pokúsime zhrnúť získané znalosti. Pri unifikovaní existujúcich metodík a navrhovaní generickej metodiky vývoja multi-agentových aplikácií sme museli prekonať niekoľko veľkých problémov.

Prvým z nich bola rôznosť použitých modelovacích jazykov jednotlivými metodikami. Jeho riešením sa ukázalo byť vyjadrenie jednotlivých metodík pomocou meta-modelovacieho jazyka SPEM. Viacero expertov na multi-agentové systémy pracovalo, a mnohí ešte stále pracujú, na vyjadrení niektorej konkrétnej metodiky meta-modelovacím jazykom SPEM. Občas však aj oni neodhadli úplne správne granularitu stereotypov profilu SPEM, respektíve vychádzali zo slovníka určitej metodiky alebo staršieho názvoslovia RUP, a nie zo slovníka SPEM. To viedlo k ich „SPEMovským“ vyjadreniam metodík (napríklad metodiky Gaia), v ktorých sa napríklad nerozlišuje medzi pojmom „fáza“ a „disciplína“, alebo sa zvolil nesprávny stupeň granularity vyjadrenia a pojmy „krok“, „aktivita“, „definícia činnosti“ boli použité v nesprávnej úrovni. S týmto problémom sme sa vysporiadali tak, že pri práci s takto vyjadrenou „spemovskou“ metodikou sme porovnávali a kontrolovali správnosť použitia jej stereotypov profilu SPEM s definovanými stereotypmi profilu SPEM organizáciou OMG.

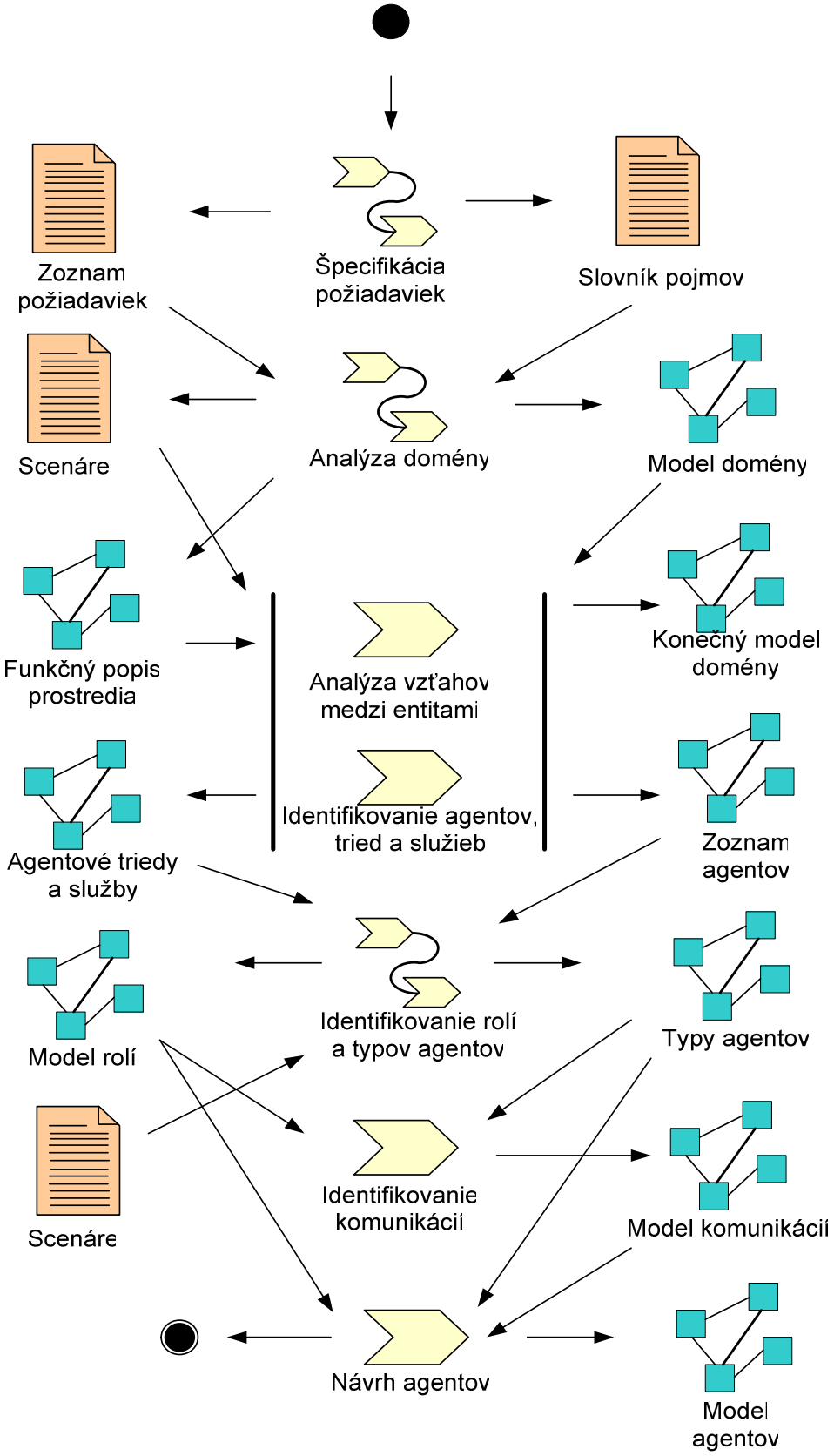
Druhým problémom bola metodikami použitá rôzna úroveň jemnosti spracovania v rámci definícií činností – ako pojednávame v kapitole 3. Niektoré metodiky neuvažujú napríklad o disciplíne zhromažďovania požiadaviek, iné zasa uvažujú až o dvoch disciplínach – o disciplíne zhromažďovania skorých a disciplíne zhromažďovania neskorých požiadaviek. Vyriešili sme to porovnávaním metodík na ešte jemnejšej úrovni definícií činností – na úrovni aktivít a krokov. V rámci najviac skúmaných metodík, metodík Gaia, Adelfe a Passi, sme identifikovali takmer 100 aktivít alebo krokov (viď príloha A) vyskytujúcich sa v niektorých z ich disciplín zberu požiadaviek, analýzy, alebo dizajnu. Tieto aktivity a kroky, spolu s nimi vytváranými výstupnými artefaktmi, nám pomohli navrhnúť granularitu pre generickú metodiku uvedenú nižšie. Rovnako nám pomohli urobiť kontrolu navrhnutej metodiky vzhľadom na podrobnejšie skúmané metodiky Gaia, Adelfe, Passi. Nie je

možno úplne korektná, niektoré aktivity môžu byť jednokrokové a preto by možno mali byť označené ako kroky, nepovažujeme to však za závažnejší problém.

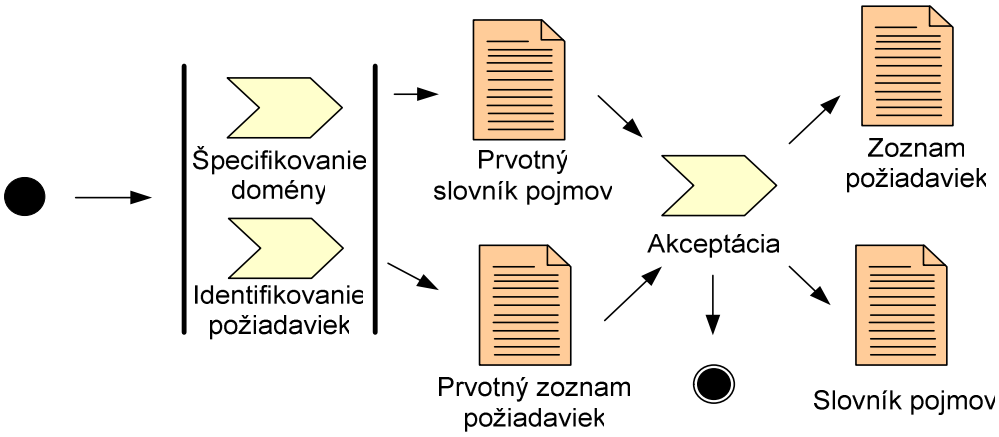
Tretím problémom bolo rozdelenie novej metodiky na disciplíny. Vynárala sa otázka, ktoré aktivity či niektoré kroky patria ešte do disciplíny zberu požiadaviek, alebo už do disciplíny analýzy, resp. podobne na rozhraní analýzy a dizajnu. Rozdelenie aktivít do disciplín nájdete v kapitolách 4.3 až 4.5.



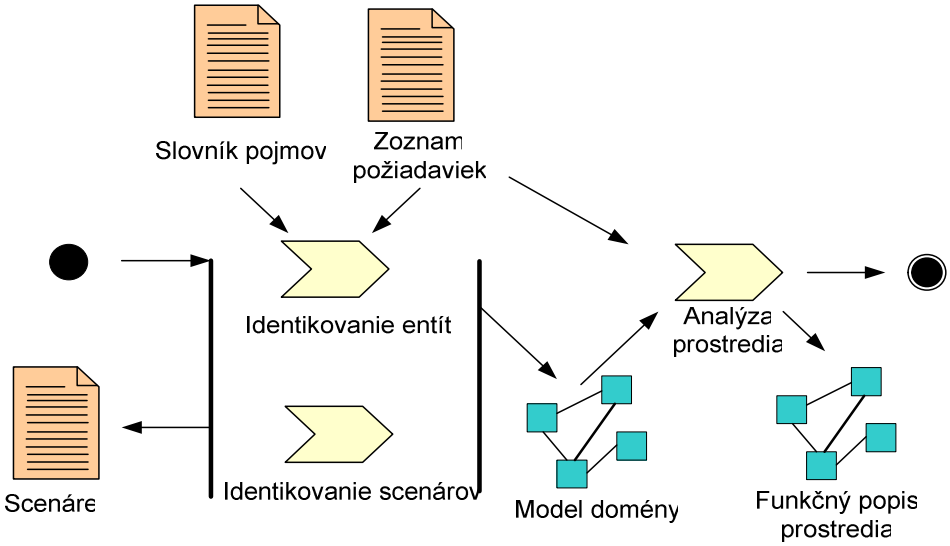

*Obr. č. 33: Kompletný proces navrhnutej metodiky*



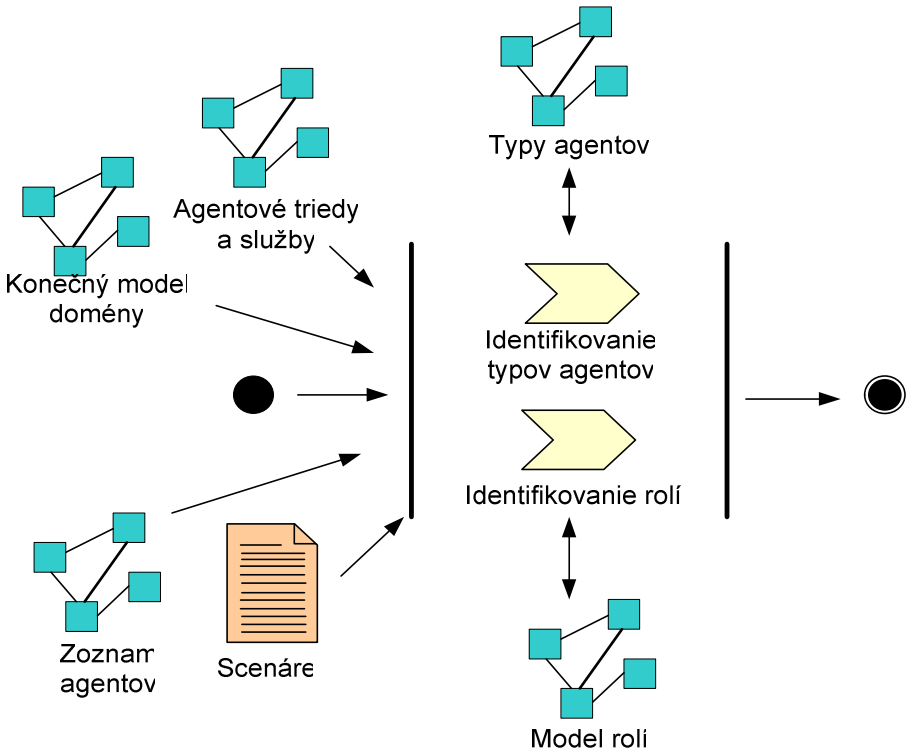

*Obr. č. 34: Celkový model definícií činností a produktov činností navrhnutej metodiky*




*Obr. č. 35: Definícia činnosti Špecifikácia požiadaviek*




*Obr. č. 36: Definícia činnosti Analýza domény*




*Obr. č. 37: Definícia činnosti Identifikovanie rolí a typov agentov*

## 4.1 Prehľad a popis krokov, aktivít a definícií činností navrhnutej metodiky

Definícia činnosti: **Špecifikácia požiadaviek**

Cieľ: Táto aktivita sa zameriava na popísanie prostredia a vyvíjaného systému. Zainteresované osoby (koncoví užívatelia, analytici, dizajnéri,...) zostavia zoznam funkčných a nie-funkčných požiadaviek, čím špecifikujú, aký systém je potrebné vytvoriť. Cieľom je identifikovať požiadavky.

Vstupné artefakty:
Výstupné artefakty: zoznam požiadaviek, slovník pojmov

Aktivity:

1. Identifikovanie požiadaviek: Vytvorenie zoznamu funkčných a nie-funkčných požiadaviek.
2. Špecifikovanie domény:

3. Akceptácia: Zoznam požiadaviek musí byť schválený aj zákazníkom, aj analytikom.

---

Aktivita: **Identifikovanie požiadaviek**

Cieľ: Cieľom je identifikovať funkčné a nie-funkčné požiadavky požadované od vyvíjaného systému. Zainteresované osoby (koncoví užívatelia, analytici, dizajnéri,...) zostavia prvotný zoznam funkčných a nie-funkčných požiadaviek.

Vstupné artefakty:
Výstupné artefakty: prvotný zoznam požiadaviek, prvotný slovník pojmov

Kroky: 1. Definovanie užívateľských požiadaviek

---

Aktivita: **Špecifikovanie domény**

Cieľ: Cieľom tejto aktivity je z dokumentov popisu procesu a popisu cieľov vytvoriť špecifikáciu domény

Vstupné artefakty:
Výstupné artefakty: prvotný zoznam požiadaviek, prvotný slovník pojmov

Kroky: 1. Popíš doménu

---

Aktivita: **Akceptácia**

Cieľ: Táto aktivita má za úlohu potvrdiť identifikované požiadavky akceptáciou zainteresovaných strán. Po akceptácii máme k dispozícii slovník pojmov a zoznam požiadaviek.

Vstupné artefakty: prvotný zoznam požiadaviek, prvotný slovník pojmov
Výstupné artefakty: slovník pojmov, zoznam požiadaviek

Kroky: 1. Potvrď platnosť požiadaviek, validácia

---

Definícia činnosti: **Analýza domény**

Cieľ: Cieľom tejto aktivity je analyzovať slovník pojmov a zoznam požiadaviek, pomocou nich identifikovať entity a vytvoriť funkčný popis prostredia a model domény.

Vstupné artefakty: zoznam požiadaviek, slovník pojmov
Výstupné artefakty: model domény, scenáre, funkčný popis prostredia

Aktivity/kroky:

1. Identifikovanie entít: Analytik identifikuje entity ovplyvňujúce systém, aj aktívne, aj pasívne a zaznamená ich do prvotného modelu domény a prostredia.
2. Identifikovanie scenárov: Identifikácia všetkých sekvencií interakcií medzi aktormi a systémom a identifikácia ich vstupov a výstupov.
3. Analýza prostredia: Analytik preskúma vlastnosti prostredia. Výstupom je rozšírený model domény a prostredia.

---

Aktivita: **Identifikovanie entít**

Cieľ: Analytik v tejto aktivite identifikuje všetky aktívne a pasívne entity a uloží ich do modelu domény a prostredia.

Vstupné artefakty: zoznam požiadaviek, slovník pojmov
Výstupné artefakty: model domény a prostredia

Kroky: 1. Zisti entity
2. Definuj koncepty

---

Aktivita: **Identifikovanie scenárov**

Cieľ: Identifikácia všetkých sekvencií interakcií medzi aktormi a systémom a identifikácia ich vstupov a výstupov.

Vstupné artefakty: zoznam požiadaviek, slovník pojmov

Výstupné artefakty: scenáre

Kroky: 1. Definuj vstupné a výstupné podmienky, vstupy a výstupy
2. Odhaľ súvislosti, limity a obmedzenia

---

Aktivita: **Analýza prostredia**

Cieľ: Analytik preskúma vlastnosti prostredia. Výstupom je funkčný popis prostredia.

Vstupné artefakty: zoznam požiadaviek, model domény
Výstupné artefakty: funkčný popis prostredia

Kroky: 1. Charakterizuj prostredie
2. Spíš use-case diagramy

---

Aktivita: **Analýza vzťahov medzi entitami**

Cieľ: Analýza interakcií a súvislostí medzi entitami vyskytujúcimi sa v systéme.

Vstupné artefakty: scenáre, model domény a funkčný popis prostredia
Výstupné artefakty: konečný model domény, zoznam agentov

Kroky: 1. Identifikuj interakcie

---

Aktivita: **Identifikovanie agentov, tried a služieb**

Cieľ: Pozostáva z vyjasnenia funkcionality každého agenta. Pre každého agenta sa pomocou atribútov špecifikuje úroveň jeho vedomostí.

Vstupné artefakty: model domény, funkčný popis prostredia, scenáre
Výstupné artefakty: zoznam agentov, agentové triedy a služby

Kroky: 1. Identifikuj agentov
2. Identifikuj triedy
3. Identifikuj služby

---

Definícia činnosti: **Identifikovanie rolí a typov agentov**

Cieľ: Pozostáva z preskúmania možností vývoja systému za účelom identifikovania vonkajších prejavov správania agentov. Identifikujú sa všetky možné komunikačné cesty medzi agentmi. Každá z nich predstavuje scenár vzájomného pôsobenia agentov pracujúcich za účelom dosiahnutia požadovaného správania systému.

Vstupné artefakty: scenáre, konečný model domény, zoznam agentov, agentové triedy a služby
Výstupné artefakty: typy agentov, model rolí

Aktivity/kroky:

1. Identifikovanie typov agentov
2. Identifikovanie rolí

Poznámka: Poradie týchto dvoch aktivít nie je pevne určené. Podľa rozsahu poznatkov o vyvíjanom MAS môže byť ľahšie napríklad najprv identifikovať typy agentov a podľa nich určiť role, ktoré plnia (príkladom sú futbalisti = stopéri, útočníci,...). Druhom možnosťou je ľahšie identifikovanie rolí (ako to je napr. v metodike Gaia) a až potom odhalenie typov agentov (príkladom sú mravce = v rámci spoločenstva plnia roly, jednotlivé typy identifikujeme neskôr).

---

Aktivita: **Identifikovanie typov agentov**

Cieľ: Cieľom je pomocou identifikovaných rolí odhaliť typy agentov a ich ciele.

Vstupné artefakty: scenáre, konečný model domény, zoznam agentov, model rolí, agentové triedy a služby
Výstupné artefakty: typy agentov

Kroky: 1. Identifikuj typy agentov
2. Analyzuj ich vzťahy

---

Aktivita: **Identifikovanie rolí**

Cieľ: Cieľom je pomocou identifikovaných typov agentov odhaliť ich role a ich ciele.

Vstupné artefakty: scenáre, konečný model domény, zoznam agentov, typy agentov, agentové triedy a služby
Výstupné artefakty: model rolí

Kroky: 1. Identifikuj role
2. Analyzuj ich vzťahy

---

Aktivita: **Identifikovanie komunikácií**

Cieľ: Cieľom tejto aktivity je identifikovanie komunikácií a definovanie komunikačných protokolov agentov.

Vstupné artefakty: model rolí, typy agentov
Výstupné artefakty: model komunikácií

Kroky: 1. Identifikuj odosielateľa a prijímateľa
2. Identifikuj komunikáciu a protoky

---

Aktivita: **Návrh agentov**

Cieľ: Cieľom tejto aktivity je špecifikovať vlastnosti agentov, ich znalosti, vedomosti, zručnosti, vnímanie prostredia, seba a iných agentov, a proces riešenia konfliktných situácií.

Vstupné artefakty: model komunikácií, zoznam agentov, model rolí, typy agentov
Výstupné artefakty: model agentov

Kroky: 1. Definuj vlastnosti agentov
2. Definuj počet agentových inštancií
3. Vytvor agentový model

## 4.2 Prehľad a popis artefaktov navrhnutej metodiky

**Scenáre**: Dokument/model popisujúci súvislosti medzi entitami pomocou vstupov a výstupov, spolu aj so vstupnými a výstupnými podmienkami. Na reprezentáciu môžeme použiť diagramy spolupráce alebo sekvenčné diagramy, resp. k nim sémanticky ekvivalentné.

**Slovník pojmov**: Textový dokument obsahujúci kľúčové výrazy špecifické pre skúmanú doménu a vyvíjaný systém.

**Zoznam požiadaviek**: Zoznam funkčných a nie-funkčných požiadaviek špecifických pre hľadaný systém odsúhlasený zákazníkom.

**Model domény**: Model identifikovaných interakcií a súvislostí medzi entitami vyskytujúcimi sa v systéme.

**Funkčný popis prostredia**: Slúži na zachytenie funkcionality systému. Odporúčané je stanoviť use-case diagramy.

**Zoznam agentov**: Zoznam agentov je dokument alebo model, v ktorom sú obsiahnuté všetky identifikované agenty, pričom pre každého z nich je popísaná jeho funkcionalita a množina jeho vedomostí. Možnosťou reprezentácie je napríklad diagram tried.

**Agentové triedy a služby:** Pre každého agenta zo zoznamu agentov sa identifikujú služby, ktoré ponúka pre iných agentov a identifikujú sa prvé agentové triedy (napríklad pomocou diagramov tried), ktoré sú v aktivite identifikovanie rolí a typov agentov podkladom pre vytvorenie modelu rolí a typov agentov.

**Typy agentov**: Model zachytávajúci identifikované typy agentov podľa ich vonkajšieho správania.

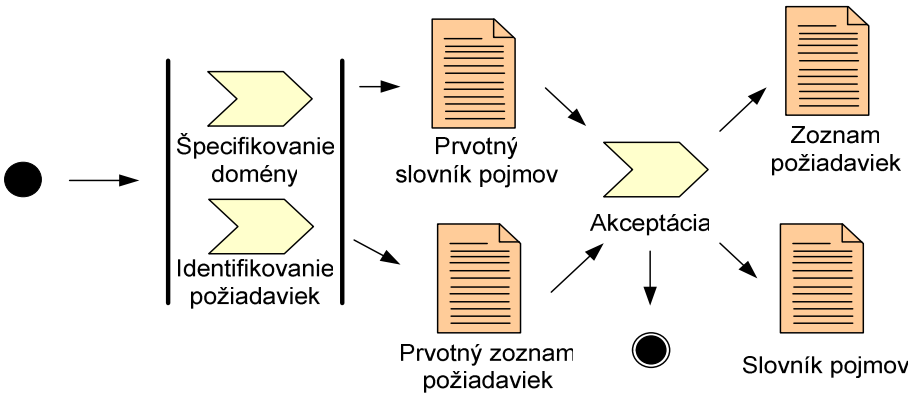
**Model rolí**: Model zachytávajúci identifikované role agentov podľa ich vonkajšieho správania.

**Model komunikácií**: Obsahuje množinu možných komunikačných ciest pre identifikované scenáre a navrhnuté komunikačné protokoly.

**Model agentov**: Model, v ktorom sú špecifikované vlastnosti agentov, ich znalosti, vedomosti, zručnosti, vnímanie prostredia, seba a iných agentov, a proces riešenia konfliktných situácií.

## 4.3 Disciplína zberu požiadaviek

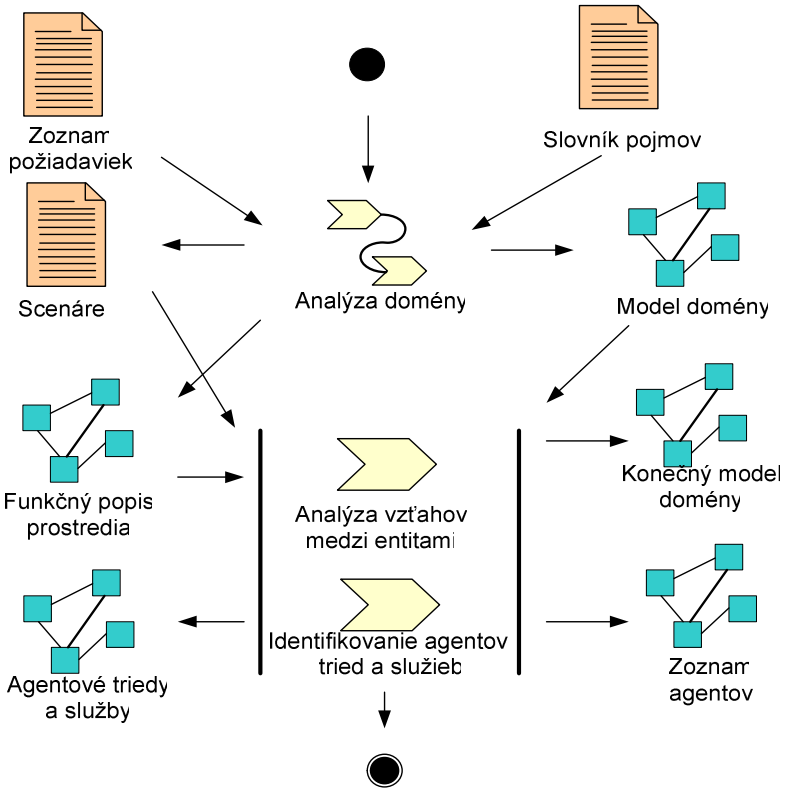
Základným cieľom disciplíny zberu požiadaviek je zhromaždenie požiadaviek kladených na vyvíjaný systém pomocou viacerých dokumentov.




*Obr. č. 38: Disciplína zberu požiadaviek*

## 4.4 Disciplína analýza

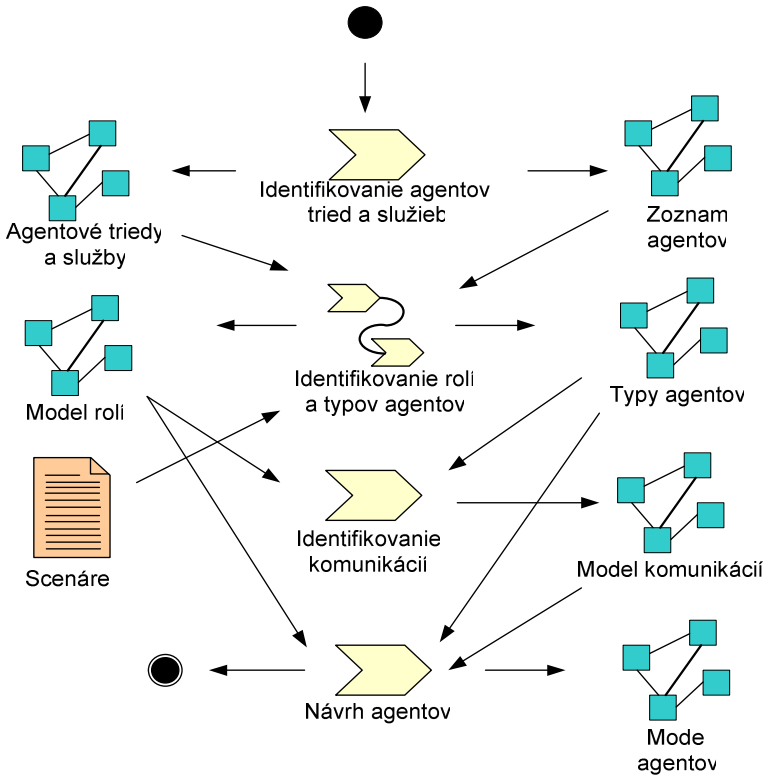
Cieľom disciplíny analýzy je potvrdenie, že sa bude analyzovať a riešiť správny problém (t.j. to čo je skutočne požadované). Po objasnení funkčných a nie-funkčných požiadaviek v disciplíne zberu požiadaviek je potrebné odhadnúť, či vlastnosti navrhovaného systému spĺňajú očakávania stakeholderov, ktorí ho budú používať. Disciplína analýza uľahčuje návrh dizajnu, pretože model analýzy poskytuje detailný popis funkcionality systému a celkový pohľad na jeho štruktúru.




*Obr. č. 39: Disciplína analýza*

## 4.5 Disciplína dizajn

Cieľom disciplíny dizajnu je definovanie riešenia problému (t.j. ako vytvoriť požadovaný systém tak aby robil to čo má robiť), popisujúc tvorbu systému spĺňajúceho všetky požiadavky a obmedzenia objavené v disciplíne analýzy. Priebeh disciplíny dizajnu pozostáva z identifikovania výpočtových komponent systému, špecifikovaním čo vykonávajú, a zriadením rozhraní medzi komponentmi.




*Obr. č. 40: Disciplína dizajn*

# 5 Záver

Cieľom tejto práce bolo preskúmať existujúce metodiky a načrtnúť všeobecnú metodiku tvorby multi-agentových aplikácií, čo sa nám podarilo. Ako väčšina diplomových prác, aj táto predkladaná práca je neúplná a ponúka priestor na ďalšie rozšírenie. O dôvodoch jej neúplnosti sme sa zmienili na viacerých miestach práce. Pri výbere témy diplomovej práce som vedel o MAS veľmi málo. Vôbec som vtedy netušil, že o návrh všeobecnej metodiky sa pokúšajú experti na agentové systémy po celom svete, či už vedeckí pracovníci na univerzitách, alebo odborníci z praxe z rôznych oblastí, a že som si vybral rozlusknúť jeden veľmi tvrdý oriešok. Odborníci neustále vylepšujú väčšinu existujúcich metodík a rozširujú ich o odhalené slabé stránky. Najprijateľnejším riešením pri výbere metodiky sa zdá byť nastavenie metodiky na konkrétny projekt a to výberom len potrebných fragmentov z existujúcich metodík a následným ich zostavením do jednej metodiky. Problémom zostáva aj neexistencia všeobecne akceptovaného modelovacieho jazyka pre MAS. Len prednedávnom (február 2005) sa podarilo firme Whitestein Technologies vypracovať agentový modelovací jazyk AML, ktorý sa možno onedlho stane celosvetovým štandardom. Pre tých, ktorí sa zaoberajú metodikami tvorby multi-agentových aplikácií, poskytuje táto práca podnetný materiál ako riešiť problémy s

tým spojené. A pre tých, ktorí sa s podobnými problémami nezaoberali, bude možno podnetom k skúmaniu týchto problémov.

Záverom môžeme povedať, že stanovený cieľ bol sčasti dosiahnutý. Boli odhalené fragmenty existujúcich procesov a načrtnuté hrubšie rysy generickej metodiky, ktorá vychádza z existujúcich metodík.

## 5.1 Použité skratky

| | |
|---|---|
| AO | agentovo-orientovaný |
| AOSE | agentovo-orientované softvérové inžinierstvo |
| A-UML | Agent UML |
| BDI | viera, zámer, úmysel (Belief, Desire, Intention) |
| FIPA | Foundation for Intelligent Physical Agents |
| UI | užívateľské rozhranie |
| MAS | multi-agentový systém |
| MOF | Meta-Object Facility |
| OMG | Object Management Group |
| OO | objektovo-orientovaný |
| RUP | Rational Unified Process |
| SE | softvérové inžinierstvo |
| UML | Unified Modelling Language – objektovo-orientovaný modelovací jazyk použitý vo viacerých OO metodikách |

## 5.2 Prehľad použitých pojmov a pojmov definovaných v meta-modelovacom jazyku SPEM

Aktivita = je špecifická Definícia činnosti a jej úlohou je popis funkcie Vykonania roly.

Závislosť = je vývojovo-špecifický vzťah medzi vývojom/postupom Modelovacích elementov

Disciplína = je balík procesov organizovaný z pohľadu jednej disciplíny softvérového inžinierstva (analýza a dizajn a pod.)

Poučenie = je Modelovací element prepojený s ďalšími hlavnými elementmi definície procesu, ktorý obsahuje ďalšie popisy ako napríklad postupy, návody, vzory a príklady pracovných produktov a pod.

Iterácia = je v hlavných črtách Definícia činnosti, ktorá reprezentuje množinu Aktivít zameriavajúcich sa na časť vývoja softvéru výsledkom ktorého je zverejnenie softvérového produktu

Modelovací element = je element popisujúci jeden aspekt procesu softvérového inžinierstva

Vykonanie roly = je modelovací element popisujúci role, zodpovednosti a kompetencie vykonaných individuálnymi Aktivitami v rámci Procesu, a je zodpovedný za určité Produkty činnosti

Fáza = abstraktnejšia Definícia činnosti ohraničená medzníkmi

Proces = kompletný popis procesu softvérového inžinierstva za pomoci použitia pojmov Predstaviteľ činnosti, Vykonanie roly, Definícia činnosti, Produkt činnosti a pridružené Poučenie.

Predstaviteľ činnosti = Modelovací element popisujúci vlastníka definície činnosti. Používa sa pre tie definície činnosti, ktoré nemôžu byť asociované s individuálnymi Vykonaniami rolí, akými sú napríklad Životný cyklus a Disciplína.

Krok = atomický Modelovací element používaný na dekompozíciu Aktivít.

Definícia činnosti = Modelovací element procesu popisujúceho vykonanie operácií a transformácií na Produktoch činnosti pomocou Rolí. Medzi typy definície činnosti patria: Aktivita, Iterácia, Fáza a Životný cyklus.

Produkt činnosti = informácia alebo fyzická entita produkovaná alebo používaná aktivitami procesu softvérového inžinierstva. Typickými príkladmi produktov činnosti sú modely, plány, kód, dokumenty, databázy a pod.

# Príloha A:

| Metodika, definícia činnosti/aktivita/krok | názov definície činnosti/aktivity/kroku | typ definície činnosti/aktivity/kroku | výstupné artefakty |
|---|---|---|---|
| Gaia, dokument špecifikovaných požiadaviek | Zoznam požiadaviek | Artefakt | zoznam požiadaviek, slovník pojmov |
| Passi,aktivita,identifikuj požiadavky | špecifikácia požiadaviek | definícia činnosti | slovník pojmov |
| Adelfe,aktivita,Vytvorenie množiny kľúčových slov | špecifikácia požiadaviek, špecifikovanie domény | definícia činnosti, aktivita | zoznam požiadaviek |
| Adelfe,aktivita,Definovanie užívateľových požiadaviek | identifikovanie požiadaviek | aktivita | slovník pojmov, zoznam požiadaviek |
| Adelfe,aktivita,Definícia požiadaviek odsúhlasených oboma stranami | akceptácia | aktivita | slovník pojmov, zoznam požiadaviek |
| Adelfe,aktivita,Validácia užívateľových požiadaviek | akceptácia | aktivita | slovník pojmov |
| Passi,aktivita,popíš doménu | špecifikovanie domény | aktivita | model domény a prostredia |
| Adelfe,aktivita,Analýza domény a štúdium architektúry | analýza domény | definícia činnosti | model domény a prostredia |
| Passi,definícia činnosti,popis ontologie domény | analýza domény | definícia činnosti | model domény a prostredia, scenáre |
| Passi,definícia činnosti,popis domény | analýza domény | definícia činnosti | model domény a prostredia |
| Adelde,krok,zisti entity | identifikovanie entít | aktivita | model domény a prostredia |
| Passi,aktivita,definuj koncepty | identifikovanie entít | aktivita | scenáre |
| Adelfe,aktivita,Extrakcia limitov a obmedzení | identifikovanie scenárov | aktivita | scenáre |
| Adelde,krok,definuj súvislosti | identifikovanie scenárov | aktivita | scenáre |
| Adelde,krok,identifikuj chyby spolupráce | identifikovanie scenárov | aktivita | scenáre |
| Gaia,aktivita,definuj vstupné podmienky | identifikovanie scenárov | aktivita | scenáre |
| Gaia,aktivita,definuj výstupné podmienky | identifikovanie scenárov | aktivita | scenáre |
| Gaia,aktivita,identifikuj vstupy | identifikovanie scenárov | aktivita | scenáre |
| Gaia,aktivita,identifikuj výstupy | identifikovanie scenárov | aktivita | scenáre |
| Passi,aktivita,identifikuj scenáre | identifikovanie scenárov | aktivita | scenáre |
| Passi,aktivita,definuj akcie | identifikovanie scenárov | definícia činnosti | scenáre |
| Passi,aktivita,definuj predikáty | identifikovanie scenárov | definícia činnosti | model domény a prostredia |
| Adelfe,aktivita,Charakterizovanie prostredia | analýza prostredia | aktivita | model domény a prostredia |

| | | | |
|---|---|---|---|
| Adelde,krok,charakterizuj prostredie | analýza prostredia | aktivita | konečný model domény a prostredia |
| Adelde,krok,analyzuj vzťahy medzi triedami | analýza vzťahov medzi entitami | aktivita | konečný model domény a prostredia |
| Adelde,krok,identifikuj triedy | analýza vzťahov medzi entitami | aktivita | Funkčný popis prostredia |
| Adelfe,aktivita,Stanovenie use-case-ov | analýza prostredia | aktivita | Funkčný popis prostredia |
| Adelfe,aktivita,Kontrola a uznanie platnosti prototypov užívateľských rozhraní | analýza prostredia | aktivita | Funkčný popis prostredia |
| Adelfe,aktivita,Vytvorenie prototypov užívateľských rozhraní | analýza prostredia | aktivita | Funkčný popis prostredia |
| Adelde,krok,spíš zoznam use-caseov | analýza prostredia | aktivita | Funkčný popis prostredia |
| Adelde,krok,vypracuj detailné sekvenčné diagramy | identifikovanie agentov, tried a služieb | aktivita | zoznam agentov |
| Adelfe,aktivita,Štúdium interakcií medzi rôznymi entitami | identifikovanie agentov, tried a služieb | aktivita | zoznam agentov |
| Adelfe,aktivita,Identifikácia agentov | identifikovanie agentov, tried a služieb | aktivita | zoznam agentov |
| Adelde,krok,identifikovanie potenciálne spolupracujúcich entít | identifikovanie agentov, tried a služieb | aktivita | zoznam agentov |
| Adelde,krok,pre každú entitu identifikovanú v aktivite charakteristika či je autonómna | identifikovanie agentov, tried a služieb | aktivita | zoznam agentov |
| Passi,aktivita,identifikuj agentov | identifikovanie agentov, tried a služieb | aktivita | zoznam agentov |
| Passi,aktivita,zoznam agentov | identifikovanie agentov, tried a služieb | aktivita | zoznam agentov |
| Passi,definícia činnosti,identifikácia agentov | identifikovanie agentov, tried a služieb | aktivita | zoznam agentov |
| Adelfe,aktivita,Opodstatnenosť použitia adaptívneho multi-agentového systému | identifikovanie agentov, tried a služieb | aktivita | Agentové triedy a služby |
| Adelde,krok,zostroj počiatočné diagramy tried | identifikovanie agentov, tried a služieb | aktivita | Agentové triedy a služby |
| Passi,aktivita,popíš služby | identifikovanie agentov, tried a služieb | aktivita | Agentové triedy a služby |
| Gaia,aktivita,vlož službu do modelu služieb | identifikovanie agentov, tried a služieb | aktivita | Agentové triedy a služby |
| Gaia,aktivita,zlúč službu | identifikovanie agentov, tried a služieb | aktivita | Agentové triedy a služby |
| Gaia,definícia činnosti,vytvor model služieb | identifikovanie agentov, tried a služieb | aktivita | Agentové triedy a služby |
| Passi,aktivita,vzťahy elementov ontologie | identifikovanie rolí a typov agentov | definícia činnosti | model rolí a typy agentov |
| Passi,aktivita,navrhni scenáre | identifikovanie rolí a typov agentov | definícia činnosti | model rolí a typy agentov |
| Passi,definícia činnosti,identifikácia rolí | identifikovanie rolí a typov agentov | definícia činnosti | model rolí a typy agentov |
| Passi,definícia činnosti,popis rolí | identifikovanie rolí a typov agentov | definícia činnosti | typy agentov |

| | | | |
|---|---|---|---|
| Gaia,aktivita,zoskup role do agentových typov | identifikovanie rolí a typov agentov | aktivita | model rolí |
| Gaia,definícia činnosti,identifikovanie rolí v systéme | identifikovanie rolí | aktivita | model rolí |
| Gaia,definícia činnosti,vypracuj model rolí | identifikovanie rolí | aktivita | model rolí |
| Passi,aktivita,analýza vzťahov rolí | identifikovanie rolí | aktivita | model rolí |
| Passi,aktivita,definícia vzťahov rolí | identifikovanie rolí | aktivita | model rolí |
| Passi,aktivita,identifikuj role | identifikovanie rolí | aktivita | model rolí |
| Passi,aktivita,popíš role | identifikovanie rolí | aktivita | model komunikácií, zoznam agentov |
| Passi,aktivita,špecifikuj úlohy | identifikovanie komunikácií, identifikovanie agentov, tried a služieb | aktivita | model komunikácií, zoznam agentov |
| Passi,definícia činnosti,špecifikácia úloh | identifikovanie komunikácií, identifikovanie agentov, tried a služieb | aktivita | model agentov |
| Adelde,krok,urči jazyky interakcií | identifikovanie komunikácií | aktivita | model komunikácií |
| Gaia,aktivita,identifikuj odosielateľa | identifikovanie komunikácií | aktivita | model komunikácií |
| Adelfe,aktivita,Štúdium jazykov interakcií | identifikovanie komunikácií | aktivita | model komunikácií |
| Gaia,aktivita,identifikuj prijímateľa | identifikovanie komunikácií | aktivita | model komunikácií |
| Gaia,aktivita,vlož.komunik.spojenie do modelu známostí | identifikovanie komunikácií | aktivita | model komunikácií |
| Gaia,aktivita,vytvor komunik.spojenie | identifikovanie komunikácií | aktivita | model komunikácií |
| Gaia,definícia činnosti, identifikuj asociované protokoly | identifikovanie komunikácií | aktivita | model komunikácií |
| Gaia,definícia činnosti,vytvor model známostí | identifikovanie komunikácií | aktivita | model komunikácií |
| Passi,aktivita,definícia stromu protokolu | identifikovanie komunikácií | aktivita | model komunikácií |
| Passi,aktivita,definuj komunikácie | identifikovanie komunikácií | aktivita | model komunikácií |
| Passi,aktivita,identifikuj komunikácie | identifikovanie komunikácií | aktivita | model komunikácií |
| Passi,aktivita,nakresli tok udalostí | identifikovanie komunikácií | aktivita | model komunikácií |
| Passi,aktivita,popíš obsah správ | identifikovanie komunikácií | aktivita | model komunikácií |
| Passi,aktivita,vylepši komunikačné vzťahy | identifikovanie komunikácií | aktivita | model komunikácií |
| Passi,aktivita,zoznam komunikácií agentov | identifikovanie komunikácií | aktivita | model komunikácií |
| Passi,definícia činnosti,popis ontologie komunikácií | identifikovanie komunikácií | aktivita | model komunikácií |

| | | | |
|---|---|---|---|
| Passi,definícia činnosti,popis protokolov | identifikovanie komunikácií | aktivita | model agentov |
| Adelfe,aktivita,Návrh agentov | návrh agentov | aktivita | model agentov |
| Adelfe,aktivita,Štúdium detailnej architektúry a agentového modelu | návrh agentov | aktivita | model agentov |
| Adelde,krok,definuj model nekooperatívnej situácie | návrh agentov | aktivita | model agentov |
| Adelde,krok,definuj talent a nadanie | návrh agentov | aktivita | model agentov |
| Adelde,krok,definuj zručnosti | návrh agentov | aktivita | model agentov |
| Adelde,krok,detailné vypracovanie diagramov tried | návrh agentov | aktivita | model agentov |
| Adelde,krok,použitie dizajnových vzorov | návrh agentov | aktivita | model agentov |
| Adelde,krok,určenie balíkov | návrh agentov | aktivita | model agentov |
| Adelde,krok,určenie tried | návrh agentov | aktivita | model agentov |
| Adelde,krok,urči znázornenie sveta | návrh agentov | aktivita | model agentov |
| Gaia,aktivita,definuj počet agentových inštancií | návrh agentov | aktivita | model agentov |
| Gaia,aktivita,vlož agentový typ do agentového modelu | návrh agentov | aktivita | model agentov |
| Gaia,definícia činnosti, vytvor agentový model | návrh agentov | aktivita | model agentov |
| Passi,aktivita,popis úloh | návrh agentov | aktivita | model agentov |
| Passi,aktivita,reprezentovanie vedomostí | návrh agentov | aktivita | model agentov |
| Passi,definícia činnosti,definícia štruktúry agenta | návrh agentov | aktivita | model agentov |
| Passi,definícia činnosti,správanie agenta | návrh agentov | aktivita | model agentov |
| Adelfe,aktivita,Rýchle prototypovanie a zlepšenie dizajnových modelov | patrí už do časti implementácie | | |

## 5.2 Lituratúra

[01] Michael Woolbridge, Nicholas R. Jennings. Intelligent Agents: Theory and Practice. The Knowledge Engeneering Review, Vol 10 (2), 1995.

[02] Stan Franklin, art Graesser. Is It Agent, or Just a Program ?: A Taxonomy for Autonomous Agents. In Proceedings of the ECAI ′96 Workshop (ATAL), Hungary, 1996, Springer-Verlag, 1997.

[03] Ing. Marek Paralič, Programové agenty a multi-agentové systémy, 1997

[04] Michael Wooldridge, Nicholas R. Jennings, David Kinny: The Gaia Methodology for Agent-Oriented Analysis and Design [2000]

[05] Franco Zambonelli, Nicholas R. Jennings, Michael Wooldridge: Developing Multiagent Systems: The Gaia Methodology [2003]

[06] Wooldridge, M., Jennings, N.R., Kinny D. "The Gaia Methodology for Agent-Oriented Analysis and Design". Kluwer Academic Press, 2000.

[07] A. Garro, P. Turci, M.P. Huget; Meta-Model Sources: Gaia; August 11th, 2003

[08] Michael Wooldridge, Nicholas R. Jennings, Franco Zambonelli: Developing Multiagent Systems: The Gaia Methodology; October 2003

[09] OMG; Software Process Engineering Metamodel (SPEM); June 6th, 2001

[10] Massimo Cossentino, Luca Sabatucci, Valeria Seidita; SPEM description of the PASSI process rel. 0.2; December 2003

[11] M. Cossentino, C. Potts: PASSI: a Process for Specifying and Implementing Multi-Agent Systems Using UML [2001]

[13] Carole Bernon, Marie-Pierre Gleizes, Sylvain Peyruqueou, Gauthier Picard; ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering

[14] IRIT/SMAC; ADELFE's Fragments - V1; 2004; http://www.irit.fr/ADELFE

[15] Gleizes Maries-Pierre, Millan Thierry, Picard Gauthier; ADELFE: Using SPEM Notation to Unify Agent Engineering Processes and Methodology; IRIT; 10/2003

[16] Fausto Giunchiglia, John Mylopoulos and Anna Perini: The Tropos Software Development Methodology [2001]

[17] Anna Perini, Angelo Susi, and Fausto Giunchiglia: Coordination specification in multi-agent systems. From requirements to architecture with the Tropos methodology [2002]

[18] John Mylopoulos; From Entities and Relationships to Social Actors and Dependencies; 19th International Conference on Conceptual Modeling; Salt Lake City, October 2000

[19] Giunchiglia, Mylopoulos, Perini; The Tropos Software Development Methodology: Processes, Models and Diagrams

[20] Kendra Hamilton, Examining the Multiagent Systems Engineering Methodology, December 2003

[21] Mark Wood and Scott A. DeLoach, An Overview of the Multiagent Systems Engineering Methodology

[22] M. Wood, S. A. DeLoach, and C. Sparkman. Multiagent system engineering. International Journal of Software Engineering and Knowledge Engineering, April 2001.

[23] Richard Evans, Paul Kearney, Jamie Stark, Giovanni Caire, Francisco J. Garijo, Jorge J. Gomez Sanz, Juan Pavon, Francisco Leal, Paulo Chainho, Philippe Massonet: MESSAGE: Methodology for Engineering Systems of Software Agents [2001]

[24] Giovanni Caire, Francisco Leal, Paulo Chainho, Richard Evans: Agent Oriented Analysis using MESSAGE/UML [2001]

[25] Giovanni Caire, Francisco Leal, Joao Rodrigues; MESSAGE: Methodology for Engineering Systems of Software Agents

[26] Giovanni Caire, Francisco Leal, Paulo Chainho, Richard Evans; Agent Oriented Analysis using MESSAGE/UML

[27] Iglesias, C., Garijo M., Gonzalez, J. and Velasco, J.R. “Analysis and Design of multiagent systems using MAS-CommonKADS”. Intelligent Agents IV: Agent Theories, Architectures and Languages, 1997, Singh, M. P., Rao, A. and Wooldridge, M.J., eds., Lecture Notes in Computer Science 1365.

[28] OMG; Software Process Engineering Metamodel Specification; November 2002

[29] Object Management Group: UML 1.5 (http://www.uml.org) [2001]

[30] DiLeo, Jonathan, et al., „Integrating Ontodologies into Multiagent Systems Engineering“, Proceedings of the Fourth International Bi-Conference Workshop in Agent-Oriented Information Systems, AOIS 2002

[31] Norbert Zagyi; Diploma Thesis: Taxonomy and Evaluation of Software Agent Platforms; 2004

[32] O. Shehory and A. Sturm. Evaluation of modeling techniques for agent-based systems. In Proceedings of the 5th International Conference on Autonomous Agents, Montreal, June 2001.

[33] G. Caire, W. Coulier, F. Garijo, J. Gomez, J. Pavon, F. Leal, P. Chainho, P. Kearney, J. Stark, R. Evans, and P. Massonet. Agent-oriented analysis using message/uml. In Proceedings of the 2nd International Workshop on Agent-Oriented Software Engineering, 2002.

[34] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems, 2000.

[35] T. Juan, A. Pierce, and L. Sterling. Roadmap: Extending the gaia methodology for complex open systems. In Proceedings of the 1st ACM Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, July 2002.

[36] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. A knowledge level software engineering methodology for agent oriented programming.

In Proceedings of the 5th International Conference on Autonomous Agents, Montreal, June 2001.

[37] M. Kolp, P. Giorgini, and J. Mylopoulos. A goal-based organizational perspective on multiagent architectures. In Intelligent Agents VIII: Agent Theories, Architectures, and Languages, 2002.

[38] J. Mylopoulos, L. Chung, and E. Yu. From object-oriented to goal-oriented requirements analysis, January 1999.

[39] J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: the tropos project. Information Systems, June 2002.