Universitat Autònoma de Barcelona

Escola Tècnica Superior d' Enginyeria

Secció d'Enginyeria Informàtica

Modelització de plantes de producció com a sistemes multiagent i implantació parcial en un cas real

Memòria del treball de recerca

Presentat per en Jonatan Trullàs i Ledesma

i dirigit per en Lluís Ribas i Xirgo

Bellaterra, 1 de setembre de 2008

Índex

1. Introducció

El problema que motiva aquest treball es pot introduir fàcilment amb un exemple, prou simple i real, que ens permeti veure com els canvis tecnològics dels darrers 30-40 anys, juntament amb les noves estratègies comercials globalitzades, afecten profundament als sistemes de producció de les petites empreses. Aquestes, sovint tenen dificultats amb la gestió del conjunt de tecnologies diverses que han anat incorporant a planta durant la seva existència, i també per adaptar-se a la necessitat d'una ràpida adaptabilitat a aquest món canviant. En aquest capítol s'introdueix l'esquema de la planta real en que s'ha aplicat el model de solució proposat en aquest treball per aquest tipus de plantes de producció. A continuació es mostren les mancances d'aquest esquema com a motivació de la recerca, s'exposen els requeriments que ha de complir la solució proposada, i les restriccions amb que es treballa. Finalment s'exposa, breument, el model teòric i metodologia utilitats per a la solució així com els avantatges d'aquesta solució.

1.1 Descripció del problema

Aquest treball ha estat motivat per la necessitat de les petites empreses de posar al dia els seus sistemes de producció per adaptar-los al funcionament cada cop més dinàmic i versàtil dels mercats. Actualment es requereix la capacitat d'una ràpida adaptabilitat de la producció pel que fa a canvis freqüents del producte fabricat i a la fabricació concurrent de diversos tipus de producte simultàniament. Una altra necessitat és la integració dels diversos subsistemes de les empreses (i de la seva informació) a la vegada que es necessita una arquitectura modular que permeti sistemes oberts i escalables. El cost de la substitució dels sistemes de planta per uns altres que s'ajustin a aquestes necessitats, o bé de la implantació d'un nou sistema integral, no el poden assumir moltes d'aquestes petites empreses, simplement perquè disposen només d'una única planta de producció que no és viable aturar o no els és viable de muntar una planta alternativa.

Això sí, moltes vegades el que sí és factible és canviar el model d'organització de la planta. Tot i això, aquests canvis també són costosos perquè, per exemple, poden suposar l'actualització de tot el software.

Aquesta dificultat de substitució del model de planta és més greu en empreses antigues de petita o mitjana dimensió, com moltes de les que es troben a Catalunya. De fet, hi ha moltes d'aquestes empreses que han fet grans inversions en equips de darrera tecnologia des dels anys 60. El cost d'incorporació d'aquesta tecnologia, necessària i molt especialitzada en processos concrets del negoci, fa que es converteixi en insubstituïble durant un període de temps superior al previst. Així conviuen sistemes i tecnologies diverses, incorporades a planta durant els

anys. Això suposa un obstacle afegit per a la seva integració degut a la heterogeneïtat d'aquests sistemes software i hardware.

Per això l'objectiu d'aquest treball ha estat iniciar una recerca de possibles solucions a aquest tipus de problema, és a dir, fer un repàs a les tecnologies, models, metodologies i plataformes existents per veure com es poden adaptar a aquest tipus d'entorn. A més, el fet que l'autor d'aquesta recerca treballi en una empresa que proporciona serveis de desenvolupament de software a alguna d'aquestes plantes de producció, ha permès accedir a un exemple de planta real. S'han aprofitat les facilitats d'accés a la instal·lació per entendre la organització del treball, el model actual del negoci, el funcionament de planta, les tecnologies i conceptes diversos del procés productiu. D'aquesta manera s'ha pogut estudiar l'adaptabilitat de les diverses tecnologies i plataformes a la planta real i, a més, han aparegut problemes i condicionants reals que potser no s'haurien plantejat en un entorn totalment experimental.

1.2 Esquema de la planta exemple

En aquest treball l'exemple real consisteix en una petita factoria metal·lúrgica [62] on el 90% de la seva producció són components metàl·lics subcontractats per terceres empreses. La subcontractació, en aquest cas, consisteix en la fabricació de components, o elements semifabricats, que seran utilitzats per les terceres empreses a l'hora de muntar els seus productes (per exemple l'estructura d'un caixer automàtic, una porta de tren, calaixos per caixes fortes o el bastidor del cablejat elèctric d'una locomotora). És a dir, cal produir una quantitat de productes, més diversos a mesura que augmenti la quantitat de clients. En cas de que el grup de clients no fos nombrós caldria poder reconfigurar el sistema de producció de manera que es pogués produir ràpidament un conjunt de productes diferents si aquest petit grup de clients canvia. El 10% restant de la producció consisteix en productes acabats que surten directament al mercat sense formar part d'altres elements (per exemple bases de panells solars). L'empresa del nostre exemple porta més de 60 anys funcionant, primer servint a clients locals i actualment produint per clients d'arreu del món (un món cada cop més competitiu). Com que l'empresa ha anat incorporant noves tecnologies al procés productiu tenim un sistema complex format per components hardware i software heterogènies com, per exemple, un sistema ERP (*Enterprise Resource Planning*) de gestió i control de la producció, eines CAD/CAM d'oficina tècnica, eines màquina pels diversos processos (làser, punxonat, tall, plegat, etc.), software i hardware de control i gestió dels magatzems automatitzats, màquines CNC (control numèric), robots, AGV (*Automated Guided Vehicle*), etc. Totes aquestes components no estan integrades en un sistema automatitzat, obert i fàcil de reconfigurar.

A Figura 1 es mostra, de forma esquemàtica i simplificada, la organització de tot aquest sistema de components software/hardware (la quantitat de recursos i eines

s'ha reduït per l'esquema). Seguint la organització tradicional (jeràrquica, rígida i centralitzada) de l'empresa, els passos principals que segueix un nou producte abans de la seva producció, a grans trets, són:

- Disseny CAD a partir dels plànols i d'informació tècnica del producte proporcionats pel client. A partir d'aquests es disposa d'una base de dades amb les dades físiques dels elements. Des de la geometria 3D a paràmetres físics com el pes específic, la superfície total neta o el centre de gravetat. El format i localització d'aquesta informació dependrà de les eines CAD que s'hagin utilitzat.
- Optimització del material utilitzat (xapes metàl·liques en aquest cas) i generació dels programes CNC (i d'altres programacions, com la dels robots manipuladors, si s'escau). Les dades introduïdes al pas anterior s'exporten a eines SW especialitzades (hi ha algunes solucions estàndard, però les màquines una mica antigues conserven aplicacions dependents del HW específic) que permeten optimitzar el consum de xapes mitjançant un *nesting* de peces per cada xapa, i generen el programa CN específic per cada eina màquina.
- Introducció dels elements que formen el nou producte i les comandes rebudes del client al sistema ERP de gestió de l'empresa.
- Introducció de l'estructura del nou producte, els processos i recursos necessaris al sistema de control de la producció. Aquest sistema està integrat amb l'ERP, però no té connexions automàtiques amb les components utilitzades als passos 1 i 2.
- Generació de les ordres de treball en funció de les comandes rebudes.
- *Scheduling* de la producció: en aquesta fase una altra eina específica s'encarrega de planificar i assignar els recursos pels processos actius dins el sistema de control de la producció en funció de prioritats, dates d'entrega i diversos criteris que poden ser modificats en funció de les necessitats d'optimitzar temps, cost, etc. En aquest cas el lligam del planificador amb el sistema ERP és simplement una exportació de les dades mínimes necessàries per poder planificar.
- Fabricació: les ordres de fabricació passen a la cadena de producció. Durant aquesta fase es consumeix matèria primera, es fabriquen nous elements que formaran part del nou producte i, finalment, s'afegeix a l'estoc el nou producte. Aquí tenim un grup divers de components que hi intervenen. Les màquines encarregades dels diversos processos (sense cap lligam amb l'ERP), el sistema robotitzat de control de magatzems (amb el mínim lligam necessari per actualitzar la base de dades d'estocs a l'ERP). Operaris humans que introdueixen manualment informació al sistema sobre l'estat del procés productiu, etc. Però, en cap cas, les ordres de fabricació intervenen de forma activa (comportant-se com a agents) en el control de les màquines (làsers, plegadores, premses) ni dels magatzems. I aquestes màquines tampoc intervenen de forma activa

informant a les ordres de fabricació de possibles incidències, saturacions o canvis d'estat en la peça fabricada.

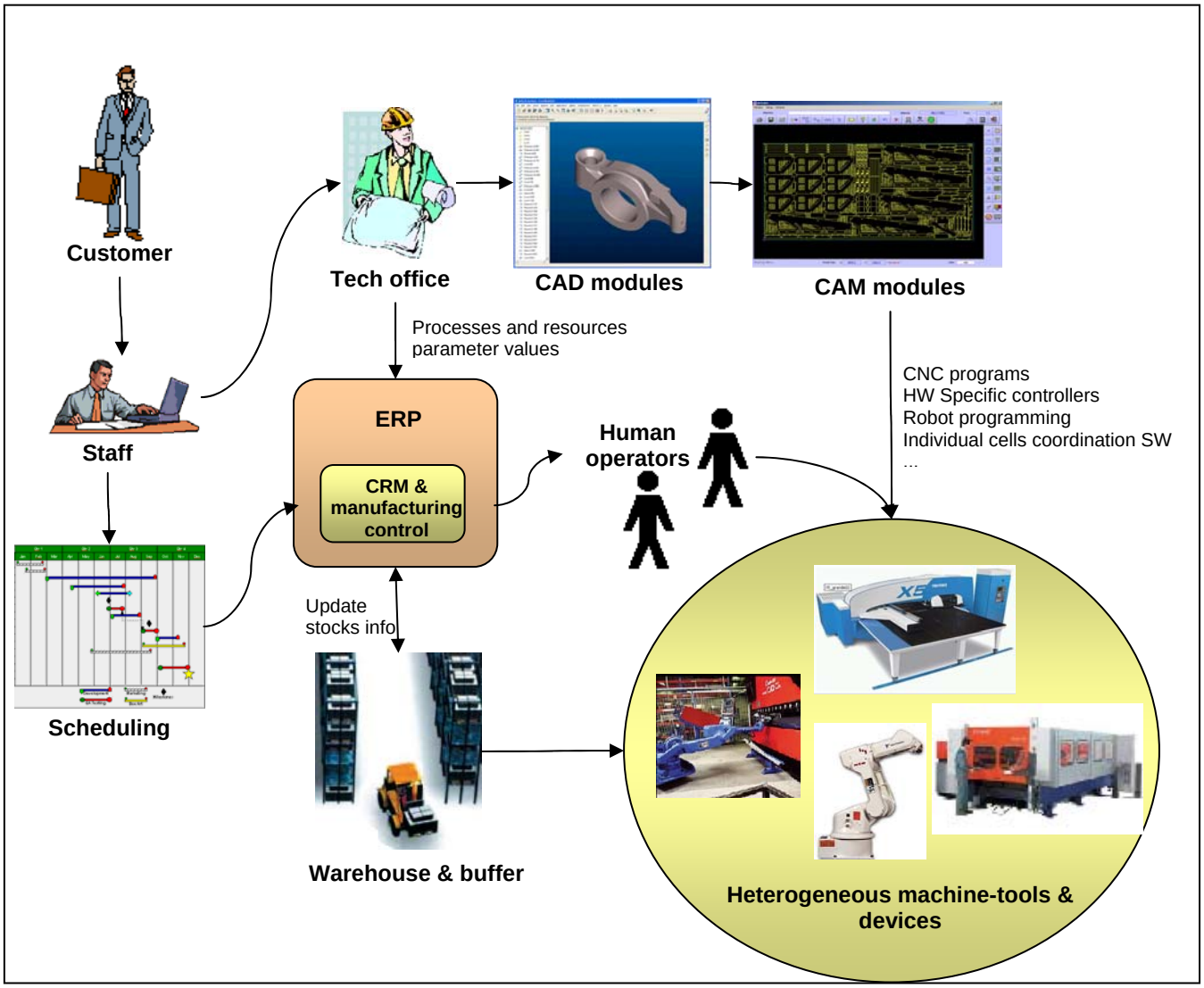


Figura 1. Esquema de la organització de la planta exemple [74]

1.3 Motivació: mancances i problemes a planta

Un cop vist l'exemple, les primeres conclusions importants que ens portaran cap a una anàlisi més profunda, i una revisió de l'estat de l'art, es poden resumir en els punts següents:

- Manca una integració a nivell d'empresa dels diversos subsistemes. Fixem-nos, per exemple, que manca una connexió forta entre les fases de disseny CAD i d'optimització de material (bàsicament eines CAD/CAM d'oficina tècnica) amb el sistema ERP. Hi ha un coll d'ampolla en el temps que necessiten els operadors humans per entrar la mateixa informació als dos sistemes. El subsistema de control del procés productiu tampoc interacciona directament amb les maquines eina en temps real. Aquesta

mancança d'integració global ens porta a una mancança de dades en temps real i centralitzades que permetin analitzar i reaccionar ràpidament a possibles imprevistos.

- Es treballa en un entorn heterogeni. La incorporació al llarg del temps de diverses tecnologies especialitzades en àmbits molt concrets de la producció fa que es disposi de mòduls HW i SW molt diversos. Tenim màquines CNC, robots de darrera tecnologia, AGV autònoms, magatzems robotitzats amb HW de control específic, bases de dades distribuïdes del ERP i els sistemes CAD/CAM, llenguatges i protocols de comunicacions diversos, maquines eines diverses, etc. Això suposarà una dificultat afegida per resoldre el problema de la manca d'integració.
- El sistema ha de ser obert i escalable. Tenint en compte que volem aconseguir la integració esmentada anteriorment i que l'empresa contínuament ha d'incorporar subsistemes heterogenis, hem de poder integrar de forma dinàmica nous subsistemes sense grans complicacions (no s'hauria de reiniciar el sistema sencer, ni aturar la producció, i en un temps raonablement curt).
- Estructura excessivament jerarquitzada i poc flexible. Es depèn del mòdul de *scheduling* per executar el procés de producció. Aquest fa les assignacions de recursos, s'escull el millor pla a seguir en funció de les restriccions i el passa al sistema de control de la producció. Finalment, el control de la producció passa les tasques a les cues de processos de les màquines (o les *cells*). Els encarregats de producció hauran de gestionar possibles imprevistos que alteren el pla global (averies, feines extra, encàrrecs de darrera hora, modificacions als programes de màquines,...), demanar materials al magatzem automàtic, etc. Com que aquest tipus d'empresa ha de ser totalment flexible, aquests imprevistos són freqüents, no hi ha una cadena única d'un sol producte que ven la pròpia empresa. Aquí una planificació a llarg termini és inviable. Per això un control distribuït, amb presa de decisions locals, però amb una coordinació global milloraria considerablement l'agilitat del sistema. Per exemple, les eines màquina, els robots i els *buffers* es podrien coordinar per assolir els objectius que els proposen les ordres de fabricació.
- Tolerància a errades. Tal i com hem vist al punt anterior, en una estructura jeràrquica qualsevol imprevist o coll d'ampolla pot col·lapsar tot el sistema. De la mateixa forma la no integració de tot el sistema pot produir dades duplicades i contradictòries entre elles. El nou plantejament que busquem hauria de solucionar aquest aspecte.

1.4 Requeriments no funcionals del problema

La solució a les limitacions i problemes mostrats anteriorment estarà condicionada per les restriccions imposades per la planta real. En aquest cas es tracta d'una

empresa petita, amb recursos limitats, i es busca una solució per una planta de producció real (no una planta model) operant dins del mercat real. Com que aquesta és una situació freqüent, el model de solució proposat pot ser aplicable a altres plantes en la mateixa situació.

Així doncs, la principal restricció a la que cal adaptar el model de solució consisteix en el fet que no es pot aturar la planta per implantar (o implementar) el model de solució proposat. Encara més, cal evitar al màxim pertorbar les operacions a planta. Això vol dir que no podem, simplement, aturar la planta i reemplaçar el conjunt de subsistemes per un nou sistema que implementi el model de solució proposada. En comptes d'això, la solució proposada ha de seguir una metodologia incremental, és a dir, sense aturar el sistema complert de planta s'aniran adaptant subsistemes (o parts del sistema antic) per etapes. Per tant, l'esquema antic de planta es transformarà progressivament "mapant" parts d'aquest cap a la part equivalent del model teòric que es proposa com a solució. Tot i que aquesta és una solució menys ambiciosa que el disseny i implementació d'un sistema nou, permet millorar progressivament el rendiment del sistema complert de planta a mesura que es va transformant en el nou model. Cada vegada que una part de l'esquema antic s'adapta al nou model s'observa una millora en el rendiment de la planta (ja sigui en temps, eficiència, capacitat de reacció, etc.).

Un altre requeriment implícit a aquesta transformació "en calent" i progressiva del sistema és la necessitat d'integració dels subsistemes llegats (*legacy systems*) amb la part del sistema que ja ha estat transformada. Degut a la naturalesa heterogènia d'aquests, el model de solució ha de proporcionar mecanismes per embolcallar (*wrapping*) o bé fer d'interfície (*translators*) entre el sistema llegat i les altres components del sistema.

A més, apareix la necessitat d'una capa de comunicacions global a planta, amb una representació estandarditzada de la informació que s'enviaran entre ells els subsistemes integrats mitjançant un llenguatge de comunicacions estàndard. Aquesta representació estandarditzada ha de contemplar un format de missatges estàndard que permeti expressar qualsevol succés dins del sistema, així com tot tipus d'informació rellevant dins del domini d'aplicació. Per això ha de ser un llenguatge obert, que permeti incloure nous conceptes relacionats amb el domini. A més, més enllà de la sintaxi del llenguatge (i la seva estructuració), la possibilitat d'associar una semàntica al llenguatge (tot i que no és imprescindible en una primera fase) pot permetre augmentar les capacitats deliberatives dels subsistemes.

Finalment, tal i com s'ha mostrat dins de les mancances de l'esquema actual de planta, l'escalabilitat del sistema heterogeni ha de permetre incloure nous elements al sistema i integrar-los sense dificultat. Aquestes components afegides han de poder informar de la seva existència, oferir les seves capacitats i cercar altres elements actius al sistema de planta. Aquesta incorporació "activa" de les components del sistema, prenent la iniciativa a l'hora d'incorporar-se al sistema,

juntament amb la mancança d'un control distribuït amb presa de decisions locals, ens porta a adoptar una solució en que els diferents mòduls del model teòric que adoptem per a la planta mostren un comportament "proactiu". Poden prendre decisions i actuar en conseqüència sense limitar-se a donar resposta a crides d'altres components.

1.5 Solució adoptada: l'agentificació gradual

La solució que es proposa en aquet treball es basa en la teoria i tecnologia multiagent. Els sistemes multiagent [76][37] (MAS) incorporen tots els requeriments que s'han exposat anteriorment per al model teòric de solució per a la planta. Tal i com es veurà amb més detall dins el capítol d'estat de l'art, els agents són entitats autònomes, situades en un entorn (real o físic) i tenen un comportament proactiu. Els MAS, formats per col·lectius d'aquestes entitats, integren agents que es poden comunicar i cooperar entre ells. Per això la tecnologia i recerca en aquest camp proporciona plataformes, *frameworks*, paradigmes i eines per integrar, comunicar i relacionar els agents entre sí. Es disposa de llenguatges estàndard i genèrics per a la comunicació entre agents (ACL) [49][26], *middleware* que serveix de capa de comunicacions, serveis de pàgines grogues i blanques perquè els agents ofereixin els seus serveis i es donin a conèixer quan s'afegeixen a un sistema existent. A més, els llenguatges de comunicació suporten la utilització d'ontologies. En el cas de la planta de producció la utilització d'una ontologia permet definir sense ambigüitats tots els conceptes del domini d'aplicació, les relacions entre ells, i associar-hi una semàntica. Això ens facilita les tasques d'integració de sistemes heterogenis. Finalment, com també es mostrarà al capítol d'estat de l'art, en un entorn de planta de producció com aquest tenim components físics i mòbils que també poden ser modelats com a agents. Més endavant també es mostrarà amb més detall la relació entre els sistemes multirobot i el model multiagent.

Així doncs, adoptant el model multiagent com a solució teòrica al problema de la planta de producció, en aquest treball es proposa una metodologia basada en la transformació progressiva del model antic de planta cap a un model multiagent de la planta degut als requeriments que hem vist anteriorment. Per aquest procés de transformació, que anomenem "agentificació", caldrà, de forma resumida, fer les següents tasques [74]:

- Creació d'un model multiagent de la planta: aquest serà el model teòric de la planta totalment agentificada, és una visió estructural de la planta. Durant el procés d'agentificació es seleccionaran components crítiques del model antic i es transformaran ("agentificaran") en la part corresponent al model multiagent.
- Creació o adaptació d'una ontologia d'alt nivell: tot i que aquesta no és una tasca imprescindible (la solució es pot limitar a utilitzar un llenguatge ACL estàndard per transportar continguts *ad-hoc*), és interessant revisar

repositoris d'ontologies de producció i de modelatge de productes per veure si se'n pot aprofitar o derivar alguna. En el pitjor dels casos es pot crear una ontologia pel domini d'aplicació concret, i estendre-la a mesura que s'amplia el sistema. D'aquesta manera es disposarà d'un model del coneixement associat al domini del sistema, al qual es pot associar una semàntica. Aquesta "vista semàntica" del coneixement del domini permetrà facilitar l'entesa entre subsistemes, i afegir raonaments sobre el contingut dels missatges.

- Implantació d'un *middleware* apropiat: és important fer una revisió de les plataformes i tecnologies MAS existents per veure si són adaptables com a capa de comunicacions o *middleware* entre agents al model de planta que estem tractant. Cal fer aquesta revisió per no quedar limitats amb la capa de comunicacions existent a planta. Les plataformes multiagent disponibles ja disposen d'eines que faciliten la implantació i desenvolupament dels agents. Per exemple, pàgines blanques, distribució de màquines "contenidores" d'agents, definició i utilització de llenguatges estàndard, etc. En algun cas, degut a les limitacions de la planta real, potser no serà viable a curt termini implantar una plataforma multiagent existent com a *middleware* i caldrà utilitzar la xarxa de comunicacions existent amb les adaptacions necessàries. De totes formes, si les adaptacions *ad hoc* de la xarxa es fan d'acord amb el model multiagent de la planta, la implantació (o substitució per un altre) del *middleware* no ha de representar cap problema costós ni cap coll d'ampolla.
- Implementació dels agents bàsics del sistema: com que a les fases inicials l'agentificació consistirà principalment en embolcallar i controlar els sistemes llegats mitjançant agents, caldrà dissenyar un conjunt d'agents genèrics que continguin una abstracció de les funcions o tasques genèriques que cal dur a terme normalment per integrar un sistema llegat. Normalment, per aconseguir aquesta agentificació de sistemes llegats implementarem dos tipus d'agents [58]: *wrappers*, pels casos en que es pot controlar el sistema directament mitjançant una interfície software (l'agent "embolcalla" el sistema), i *translators* per quan no es pot controlar el sistema directament, però es pot accedir a les seves entrades i sortides (per exemple detectant quan ha generat un fitxer nou, o bé s'ha modificat una base de dades). Un altre exemple serien els agents encarregats de la recerca i transformació de documents a un format estàndard, com XML que compleixi un esquema XSD [77].
- Simulació del sistema: la simulació pot ser una eina interessant en un context d'agentifiació com aquest, sobretot perquè el MAS s'està construint sobre un sistema existent i totes els canvis han de ser verificats prèviament. Tal com es mostrarà a l'estat de l'art, la simulació ajuda a les tasques de verificació, disseny i presa de decisions sobre el sistema. La simulació, però, no és un objectiu per si mateixa, per tant, com a eina, cal tenir clar quin nivell de detall o mòdul es vol simular i no ha de suposar un

cost considerable, especialment en el cas de la simulació de plantes de producció.

- Adaptació, reutilització i implementació d'agents específics: més enllà dels agents genèrics per embolcallar els sistemes existents, un dels propòsits de transformar el model antic en un MAS és permetre la incorporació de nous elements al sistema. Per això, un cop agentificats els sistemes llegats, potser caldrà implementar nous agents específics per al sistema, amb noves tasques assignades, no dedicats a integrar sistemes existents. Un exemple d'això seria la implementació d'agents que representin les ordres de fabricació a planta, més enllà dels agents que embolcallen el mòdul de control de la producció. També cal analitzar la reutilització d'agents similar d'altres sistemes o llibreries.

1.6 Utilitats i avantatges de la solució proposada

Els beneficis de la transformació de la planta en un MAS, com s'ha introduït anteriorment, són prou clars a nivell genèric, és a dir, per a qualsevol planta similar a la de l'exemple on el sistema implementa la tecnologia multiagent. En una planta completament agentificada, el fet de disposar d'un model del sistema complert, amb una clara distribució dels rols i processos de la planta, permet fer fàcilment canvis en el model de negoci. De fet, l'agentificació ens ha proporcionat una estructura de la planta, una base sobre la qual es poden analitzar els possibles canvis organitzatius o de negociació. Per exemple, si els agents del sistema negocien directament entre ells, prenent decisions totalment distribuïdes, hi haurà una certa dificultat a l'hora de fer una planificació global o un control dels objectius globals del sistema. En aquest cas, es podria implementar un sistema holònic de fabricació [30][33] per donar-li una certa organització jeràrquica flexible de control. Per fer això només caldria adaptar els agents perquè es comportessin com a holons. En altres casos, potser es podria decidir aplicar altres tècniques d'intel·ligència distribuïda per als agents (físics i software). En qualsevol cas, el resum és que el sistema MAS és l'estructura sobre la qual es pot "jugar" per centrar els beneficies en un o altre sentit.

De totes formes, tot això són beneficis d'un sistema completament agentificat, i en aquest treball, degut a les restriccions de la planta, es proposa una solució progressiva. Per tant, tenint en compte que el procés pot ser llarg i, en alguns casos, difícilment s'arribarà a tenir un MAS complert de la planta, ens hem de preguntar quins beneficis aporta aquesta solució menys ambiciosa. A més, al treballar sobre un exemple real, potser no sempre es pot seguir la metodologia proposada de forma estricta. D'aquí que el principal avantatge d'aquesta metodologia sigui el fet que les millores al sistema es facin present ja des de les primeres fases del procés d'adaptació. I això, a més de la confiança de l'equip directiu de la planta (un avantatge "polític"), representa els següents beneficis:

- Facilitat d'interpretació dels nous sistemes i dels llegats: el procés d'integració d'un subsistema implica recuperar informació que potser ja s'havia perdut. En ocasions hi ha màquines eina que fa anys que operen controlades per un software específic que només sap utilitzar un conjunt reduït d'operaris. Per tant, l'agentificació implicarà recuperar el coneixement de les capacitats i operativa del mòdul. D'aquesta forma tindrem una abstracció o interfície externa del seu comportament.
- Detecció de parts crítiques: el model de la planta agentificada que s'obté a la primera fase del procés d'agentificació permet, a més, distingir quines són les parts del sistema que afecten més al rendiment global. Es podria tractar de funcions del model de negoci no assignades a cap agent, o bé mancança de comunicacions entre agents del model, etc. Sense tenir aquest model teòric no es té una visió global de les funcions, rols i assignació d'aquests als diferents mòduls. Per tant, es tracta de detectar les parts d'aquest en les quals es concentra el flux de dades, pas de missatges, o bé estan assignades a recursos (màquines, operaris humans, software,...) limitats pel que fa a nombre, velocitat o bé serveis que ofereixen. Per exemple, si el model és a nivell funcional de processos de planta [63], processos que fan de "pont" entre dos subsistemes, amb tasques assignades a operaris humans, són candidats a ser agentificats. En aquest mateix tipus de representació, la disponibilitat de diversos rols per a una mateixa entitat de planta permet plantejar la possibilitat (no sempre millor) d'implementar algun dels rols en un sistema físic separat, amb comunicació explícita entre ells, etc. També es pot detectar que algunes entitats del model teòric mantenen un gran nombre de "diàlegs" i negociacions entre elles per acordar algun tipus de serveis, cosa que es podrien estalviar incorporant agents coneixedors de quina entitat ofereix cada servei (pàgines grogues) per estalviar aquestes comunicacions. En qualsevol cas, les parts crítiques detectades són les principals candidates a ser agentificades en primer lloc
- Ràpida millora en rendiment del sistema: abans ja s'ha fet referència a aquest aspecte, que és conseqüència de l'anterior (detecció de parts crítiques) Com que s'intervé sobre la planta en funcionament, i el model multiagent de la planta ens permet detectar els punts crítics, el primer que es fa és integrar camins de dades redundants (informació entrada dues vegades) o inexistents (en funció de les sortides d'un mòdul operaris humans operaven en un altre mòdul, ara això ho fan els agents). Per exemple, parts crítiques per a les quals s'ha detectat una mancança de comunicacions amb altres entitats del sistema poden donar resposta més ràpida si s'implementen nous canals de comunicació.
- Major facilitat de monitorització dels subsistemes: la proactivitat dels agents permet sol•licitar-los només la informació que pot ser interessant i ells s'encarregaran de preparar-la i servir-la si és necessari. Això és especialment important en màquines eina relativament antigues i de les

quals no es disposava d'un software de control amigable pels humans i fàcilment accessible.

1.7 Organització d'aquesta memòria

Un cop introduït el problema mitjançant l'exemple de la planta real, requeriments específics inclosos, i s'ha proposat l'adopció d'una solució concreta, els capítols següents mostren de forma organitzada les tasques que s'han hagut de dur a terme per realitzar aquest treball.

En primer lloc, el capítol 2 recull els conceptes, models i paradigmes més importants que s'han recollit fent una revisió a l'estat de l'art. És un recull, a nivell teòric, dels conceptes que poden ser incorporats a les possibles solucions que es plantegin per aquest problema. Per això es repassa la teoria en camps tan diversos com el control de robots, sistemes de producció reconfigurables, sistemes multiagent, llenguatges de comunicacions i eines de simulació.

Si una de les conclusions del capítol 2 és la idoneïtat del model multiagent com a base per al model de solució, el capítol 3 entra al terreny pràctic per mostrar algunes plataformes multiagent existents, i que poden ser candidates a ser implantades dins de la solució proposada. En concret, JADE es mostra com a plataforma més utilitzada i amb més recursos de tot tipus actualment, i s'explica perquè s'ajusta als requisits de la solució que es busca. A més, tot i que actualment en desús, es mostra Zeus per veure que hi ha alternatives. Però se'n destaca que les eines software que aporta aquesta plataforma són només una part de Zeus, són el suport a una metodologia per al disseny i implementació de sistemes multiagent. Aquesta metodologia es pot aplicar amb altres plataformes.

Entrant a la part pràctica del treball, s'ha hagut de crear el model teòric de la planta, després d'una anàlisi de la planta real. El capítol 4 mostra aquest model a nivell funcional. Es destaquen els processos principals de la planta, les seves interaccions, i les possibles entitats que hi intervenen.

El capítol 5 mostra el procés d'implantació parcial a la planta real d'una part del model teòric. S'expliquen les correspondències amb el model teòric, els condicionants a la seva implantació a planta, el model de comunicacions, el model d'informació estandarditzada, i els mòduls finalment implementats.

Finalment, les conclusions i els possibles camins que s'obren a nivell de recerca es comenten al capítol 6. Un petit annex conté la llista d'acrònims utilitzats en aqueta memòria, just abans de la bibliografia.

2. Estat de l'art: el model teòric

2.1 Introducció

En aquest capítol farem un repàs als models teòrics i conceptes actuals que ens permetran adaptar antics models de producció com el de la introducció d'aquest treball als requeriments dels sistemes de producció de nova generació (o, com a mínim, respondre d'una manera àgil als problemes plantejats). A l'hora de fer aquest repàs a l'estat de l'art s'ha tingut en compte que el problema que ens ocupa és multidisciplinari, per tant caldrà buscar en diversos camps com la robòtica, la informàtica industrial, la intel·ligència artificial, les xarxes o l'electrònica

2.2 Breu perspectiva històrica

Tot i que la informatització es va començar a introduir a la indústria aproximadament el 1960, el primer impacte tecnològic es va produir a nivell d'oficina amb els sistemes de gestió, facturació, comptabilitat Més endavant aquesta informatització s'estendrà a altres aspectes de la producció fora de la oficina . De fet, fins els anys 60 les màquines i el seu control eren absolutament mecàniques (el desenvolupament del Control Numèric al MIT data del 1952), fins l'aparició dels primers sistemes de control numèric per ordinador (CNC) . Alguns autors parlen d'una època "pre-CNC" [50], caracteritzada per una competitivitat local que empenyia a centrar els esforços en una reducció de costos mitjançant l'ús de components intercanviables. En aquesta època no hi havia exigències de variació de productes (es feia un sol producte molt de temps) ni d'integració dels sistemes de producció. L'aparició dels CNC va suposar un gran impacte a la industria, ja que permetia un control més acurat de les màquines, millor qualitat, més variabilitat del producte i una integració més fàcil. Així doncs, aquesta innovació va facilitar l'aparició i desenvolupament durant els anys 70s i 80s de noves tècniques com els sistemes de fabricació flexibles (FMS) o els *Just-In-Time* (JIT). En les FMS, el fet de treballar amb *cells* formades per diverses màquines eina, un robot i un AGV permetia combinar la gran productivitat de les línies de producció amb la gran adaptabilitat i flexibilitat del treball de taller [1][50]. El JIT permetia eliminar el concepte d'inventari com a ideal per reduir costos.

Durant aquesta època es posen les bases d'una visió de la fabricació com a un sistema integrat per subsistemes que cobreixen totes els aspectes de la fabricació (disseny, planificació, gestió, producció, entrega, etc.). Els conceptes inicials d'aquesta fabricació integrada per ordinador (CIM) es publiquen el 1973 [36]. Així i

tot, els conceptes en que es basaven aquestes noves estratègies dels anys 70 i 80, no permetien aquesta integració completa del sistema. Aquestes tècniques de producció (FMS, JIT, lean, etc.) aconseguien l'automatització en unitats funcionals, creant "illes funcionals", però la tecnologia i els conceptes subjacents (SW i HW de control, elements de control, comunicacions, protocols, etc.) no facilitaven la comunicació entre les unitats funcionals. Aquesta tecnologia tampoc facilitava la integració i la coordinació entre els diferents subsistemes.

Així doncs, a partir dels 90, en un període caracteritzat per una creixent competitivitat global, i grans processos en tecnologies de la informació, aquestes dues característiques són les forces que impulsen els darrers canvis en la producció. Ara els productors concentren els seus esforços en respondre ràpidament al mercat amb productes d'alta qualitat, a un cost més baix i en menor quantitat que en els anys anteriors [50]. D'aquí que la recerca d'aquesta adaptabilitat ràpida a un mercat cada cop més agressiu i la necessitat d'una escalabilitat que la faciliti fan que es tendeixi cap a sistemes distribuïts, intel·ligents i capaços d'adaptar-se a diversos canvis de plans o a modificacions i ampliacions dels subsistemes que els composen.

2.3 Sistemes de producció de nova generació

Els problemes i mancances que hem vist a l'exemple introductori d'aquest treball (capítol 1.3) són els que impulsen a les empreses a canviar les seves estratègies de producció per adaptar-se a aquesta competitivitat global i a la necessitat d'una ràpida adaptabilitat. El projecte internacional *Next Generation Manufacturing Systems* (sota el programa *Intelligent Manufacturing Systems*, NGMS-IMS), coordinat pel *Consortium for Advanced Manufacturing* (CAM) [55], té com a principals objectius desenvolupar les millors idees i conceptes en sistemes de producció avançats (dins el context de la *Digital Factory*). També té l'objectiu d'integrar-los dins els NGMS. Les diferents fases de treball d'aquest projecte (el report final de la fase III es va entregar el desembre del 2005) han proporcionat definicions, algorismes i metodologies tal i com es pot veure a la Figura 2.

Dins de la *Task 1.1[55]* s'introdueixen els conceptes relacionats amb els NGMS. Només a tall d'exemple (la llista és prou llarga), els sistemes de producció del segle XXI contemplen conceptes com *Fractal company*, sistemes de producció autònoms distribuïts, plans per una tasca, noció d'agent, habilitat social, reactivitat, benevolència , racionalitat , tipus d'agents, *scheduling* distribuït autònom, aprenentatge, XML, etc.

Pel que fa als requeriments que han de tenir els NGMS, la *Task 1.2* ens els descriu [55][34]. Alguns d'ells són:

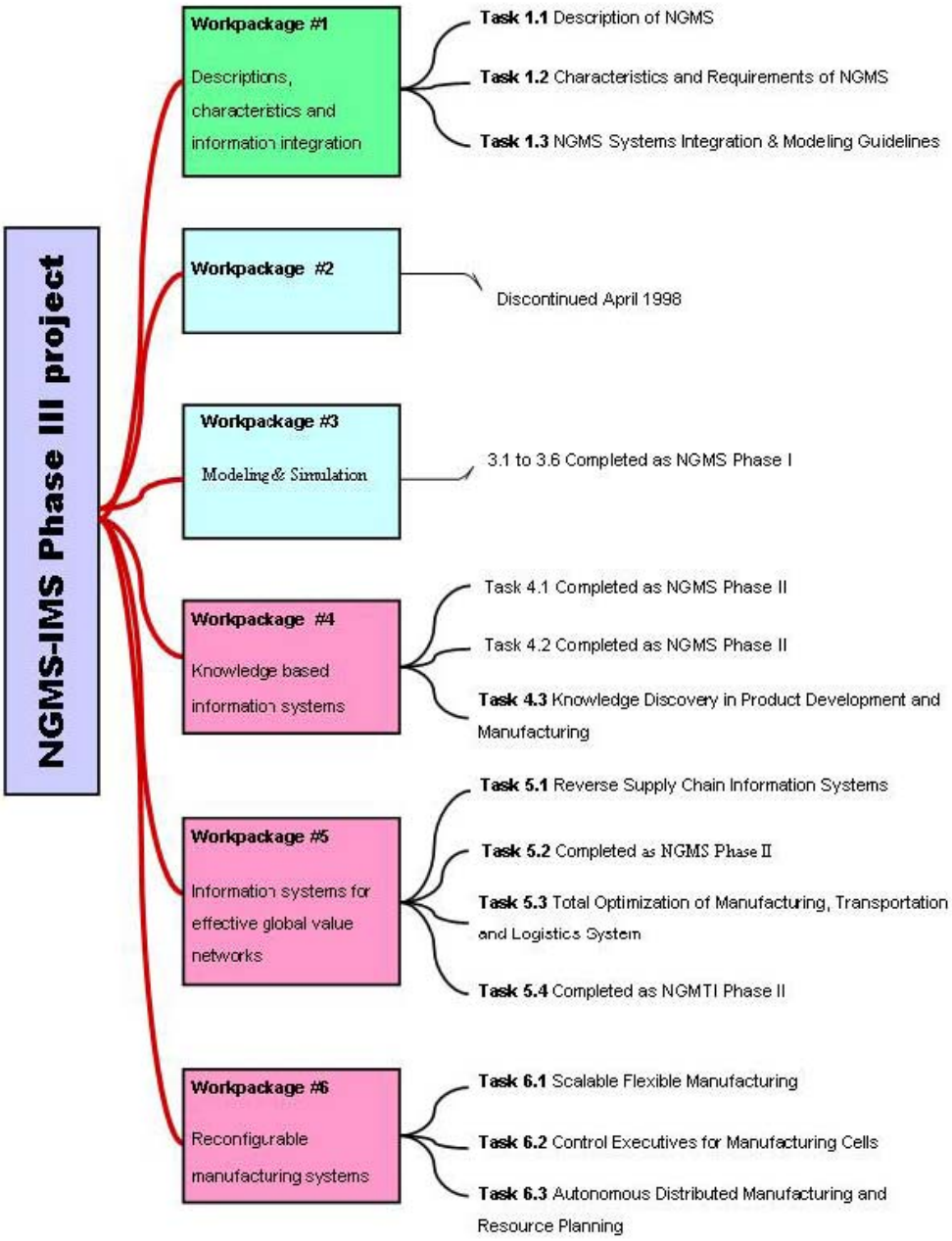


Figura 2. Àrees de treball del projecte NGMS [56]

- Integració i col·laboració a nivell dels diversos subsistemes de l'empresa. Per aconseguir-ho hauria d'estar composat d'unitats autònomes i autosuficients, capaç d'interactuar amb altres entitats del negoci i tenir interfícies consistents per poder permetre la integració entre les diverses unitats de treball.
- Integració dels humans amb el software i el hardware. Això inclou aprofitar el potencial humà, facilitar la interacció dels humans amb la màquina intel·ligent a partir d'interfícies consistents amb els processos cognitius humans,

- Ser reconfigurable , adaptable i flexible en resposta als clients. Així, perquè es pugui reorganitzar, ha de ser modular, amb una estructura adaptable ràpidament als canvis i la síntesi de les components i recursos d'un NGMS ha de ser òptima. A més ha de poder ser distribuïble (suport per *virtual enterprise*).
- Sistema adaptatiu. Capacitat de reconèixer i analitzar condicions canviants i proposar estratègies d'adaptació dintre de les restriccions i normes inherents al sistema.
- Aprenentatge. Capacitat d'incorporar coneixements adquirits de les operacions al disseny i producció del producte Representació i organització del coneixement en forma de software, de forma natural pel domini.
- Escalabilitat. Basat en estàndards i obert, de manera que els mòduls *plug and play* que implementen NGMS puguin ser fàcilment integrats, modificats i reemplaçats. A més, cal tenir en compte que ens podem trobat amb entorns heterogenis (quant a SW i HW). Aquestes components heterogènies s'haurien de poder "embolcallar" per adaptar-se a aquests estàndards.
- Tolerància a errades.

2.4 Sistemes de producció reconfigurables

Una de les propostes o paradigmes aportats pels *partners* del projecte NGMS-IMS són els *Reconfigurable Manufacturing Systems* (RMS). Els sistemes de producció usats per aquest paradigma han de ser ràpidament dissenyats, capaços de convertir-se ràpidament per a la producció d'un nou model, poder ajustar la capacitat ràpidament i integrar tecnologia nova.

Es pot definir un RMS com un sistema de producció format a partir de la incorporació de mòduls de procés bàsics (HW i SW), els quals es poden substituir, canviar o reconfigurar ràpida i fàcilment [50]. Això vol dir que, en comptes de reemplaçar-lo, podem treure elements del sistema, afegir-ne, o modificar-ne les funcions per adaptar-lo a la producció d'un nou element. Fixem-nos que el concepte de flexibilitat aquí no només fa referència a la capacitat de produir una varietat de peces (com en el cas del FSM), sinó també a la capacitat de modificar el propi sistema [50].

Així i tot, els RMS són encara una aproximació incipient, que tot just comença a sortir del camp de la recerca, i falta acabar de desenvolupar algunes de les *enabling technologies* que permetran utilitzar els RMS en els sistemes de producció de nova generació. Alguns punts encara oberts són:

- Integració de components SW i HW: tot i que hi ha moltes eines que cobreixen els diversos aspectes del a producció (CAD, CAM, control de

processos,...) cal una aproximació que permeti sistematitzar aquesta integració. A més, moltes components treballen en funció a una configuració estàtica de les màquines (per exemple sèrie o paral·lel), metres que RMS ha de permetre una configuració dinàmica i adaptable. En aquest sentit més endavant veurem com tecnologies de creixent implantació durant els darrers anys, com els sistemes multiagent o els sistemes de producció holònics són models vàlids per resoldre aquesta qüestió.

- Eines i metodologies: tenint en compte que diverses configuracions del sistema poden produir la mateixa família de peces, cal una metodologia que permeti avaluar aquestes configuracions i ajudar a la implementació del sistema que s'ajusti a la millor d'aquestes.
- Aspectes SW i HW: el conjunt de components software a baix nivell (monitorització, comunicacions,etc.) i a alt nivell (planificació, control de processos, coordinació "intel·ligent") fa que l'estructura i funcionalitat del SW de control i comunicacions sigui molt crítica i afecti directament al rendiment del sistema sencer. Des d'un punt de vista econòmic, aproximadament u 25% del cost inicial d'una eina/màquina s'atribueix del desenvolupament del SW [50]. Per definició, un RMS és extensible i reconfigurable pel fet de ser modular, però malauradament moltes màquines CNC encara contenen sistemes de control propietari, no accessibles per l'usuari o molt costosos, això fa que no suportin les característiques bàsiques de RMS. Un altre aspecte tecnològic, per exemple, és la utilització fins fa pocs anys dels PLC (*Programmable Logic Controllers*) pel control i monitorització industrial de sistemes d'events discrets. Aquests estan molt limitats des de la incorporació de sistemes de coneixement a la producció. Per sort els sistemes de control basats en PC estan revolucionant els sistemes de producció i ajuden a superar aquestes limitacions. I, per acabar aquest punt, l'altre aspecte crític són les comunicacions, fins fa poc gairebé totes sobre xarxes amb protocols propietaris, tot i que darrerament l'estandardització d'alguns aspectes com la terminologia dels missatges amb *Manufacturing Message Specification (MMS)* ajuden en la integració.
- Aspectes de disseny de màquines: es pot crear un RMS des d'uns mòduls bàsics anomenats *building blocks*. Ara bé, a més de l'avaluació de la configuració òptima en funció dels requeriments, cal avaluar l'extensió d'aquesta modularitat, és a dir, quin és el conjunt de *building blocks* que composa el sistema més eficient i optimitzat en funció dels requeriments concrets. Per suportar les característiques dels RMS es necessita desenvolupar una nova teoria per la síntesi topològica de màquines reconfigurables, optimització de les cadenes de moviment i estructura de la màquina, i l'anàlisi dels problemes associats.

2.5 Sistemes multiagent (MAS)

2.5.1 Introducció al concepte d'agent.

A la literatura no hi trobem una definició universalment acceptada, hi ha diverses definicions amb contribucions de la intel·ligència artificial, la filosofia , l'enginyeria i les ciències socials. Això fa que, en funció de l'autor i el seu camp de treball, trobem definicions més centrades en un o altre aspecte (alguns es centren en el coneixement, estat mental i intencions, d'altres en aspectes d'inspiració biològica com la capacitat d'autoconservació, etc). Així i tot, sembla haver-hi un consens en que el concepte d'autonomia hi ocupa una posició central. Aquesta diversitat d'aportacions al camp fa que algunes habilitats crítiques en algunes aplicacions no ho siguin en unes altres. Per exemple, la capacitat d'aprenentatge pot ser un atribut imprescindible segons algunes aplicacions, però innecessària en d'altres on un comportament reactiu simple és el més important i l'aprenentatge pot ser, fins i tot, indesitjable [76]. Una primera definició d'agent que podem adoptar és: una entitat que posseeix les propietats d'autonomia, habilitat social, reactivitat i "proactivitat" [76], i, intentant trobar les mínimes característiques que hauria de tenir un agent, podríem considerar que té les següents propietats (independentment de si parlem d'una entitat software o física)[68]:

- Autonomia: en el sentit de que és capaç d'operar sense la intervenció de cap altre agent (humà o artificial). Per tal de comportar-se de forma autònoma és necessari que processi algun nivell d'intel·ligència (com aconseguir-la variarà entre les implementacions).
- Localitzat (*situatedness*): això vol dir que aquesta entitat està localitzada en un entorn real o virtual. Aquest requeriment és fonamental per permetre'ns fer una distinció entre els agents i altres entitats software com mòduls o objectes.
- Comportament proactiu: és clau per distingir un agent d'un altre objecte passiu. Per mostrar aquest comportament, una entitat ha d'estar contínuament intentant percebre del seu entorn i, conseqüentment, intentant fer alguna acció basada en aquesta percepció.

Fixem-nos que aquestes propietats ens permeten fer una primera distinció entre el concepte d'agent i el d'objecte dins del model orientat a objectes. Si ens fixem en la propietat d'autonomia, aquest es pot pensar que els objectes també en tenen perquè ofereixen mètodes als altres objectes, amb una interfície ben definida, i les dades referents al seu estat intern estan encapsulades. Però els objectes no controlen quan ni com seran invocats els mètodes que ofereixen i, sempre que es requereix, aquests són executats. Per altra banda, els agents, si un altre agent els requereix l'execució d'una acció, poden decidir si la duran a terme i quan ho faran. És a dir, un requeriment d'executar una acció no assegura que l'agent l'executi. A més, el comportament proactiu i autònom fa que els agents puguin requerir informació o actuacions quan ho creguin convenient, el lligam "fort" entre

els objectes no permet aquest comportament. Una altra aspecte que els diferencia és que cada agent s'executa en el seu propi *thread* de control [76].

Però, a més de les característiques bàsiques anteriors, la màxima utilitat (o "poder") dels agents s'aconsegueix quan comencem a ampliar aquest agent bàsic amb habilitats per aconseguir tasques específiques, com:

- Habilitat social: per molts considerada una propietat fonamental. La capacitat de transmetre o rebre informació d'altres agents no constitueix per si sola una habilitat social (això es podria aconseguir indirectament manipulant l'entorn). Enlloc d'això, per habilitat social, ens referim a la capacitat dels agents de negociació i coordinació de forma dinàmica. L'habilitat social de comunicar-se pot variar molt entre les implementacions. Els agents altament socials s'han modelat per comunicar-se via llenguatges de comunicacions d'agents (ACL) específics, com KQML o FIPA-ACL.
- Reactivitat: en aquest cas cal tenir en compte que podem trobar dues interpretacions del concepte de reactivitat. En ambdues es tracta d'una resposta a un estímul. En un cas es tracta d'una resposta oportuna en el temps, això vol dir que hi pot haver un raonament simbòlic intern per donar aquesta resposta La segona interpretació consisteix en interpretar la reactivitat com una reacció del tipus estímul-resposta sense cap tipus de raonament simbòlic. Alguns autors parlen de comportament reflexiu, en aquest segon cas.
- Habilitat reproductiva: amb la intenció de produir agents realment autònoms les investigadors en intel·ligència artificial sovint inclouen aquesta característica com a clau per un agent.
- Possessió de recursos: basant-nos en la biologia, es proposa que els agents haurien de posseir els seus propis recursos i el comportament proactiu farà que els mantinguin i els utilitzin.
- Mobilitat: els agents físics i els software poden ser mòbils. En el cas dels físics s'inclouen robots i PDA, i en dels software s'implementen com a entitats que poden anar d'una plataforma a una altra. De fet s'ha de destacar que els agents mòbils software poden migrar a través d'agents físics mòbils.
- Racionalitat: a través de la racionalitat es força a l'agent a actuar segons la millor decisió possible.
- Intencionalitat: la possibilitat de crear agents amb "estats mentals" (coneixements, creences, objectius) explícits fa que parlem d'agents amb intencionalitat, és a dir l'arquitectura els permet raonar sobre el seu entorn i sobre sí mateixos.
- Adaptabilitat: als canvis de l'entorn, especialment en el cas dels agents físics.

2.5.2 Tipus d'agents

Un cop vist el concepte d'agent, els seus atributs bàsics i alguns atributs que ens permeten ampliar-ne la capacitat, anem a veure alguns tipus d'agents en funció dels seus atributs. A més, cal tenir en compte que podem trobar arquitectures híbrides en les quals l'agent està composat per subsistemes que presenten característiques d'un o altre dels tipus aquí esmentats.

- Agents reactius: el sistema d'arbitratge entre les percepcions i les accions és un mecanisme estímul/resposta. Com hem vist abans, en el cas més estricte no hi ha cap raonament simbòlic ni memòria d'estat. Des d'un punt de vista de la intel·ligència artificial clàssica es poden considerar "muts" degut a la seva mancança de raonament [68]. Així i tot, el seu estudi és interessant degut a la possibilitat d'utilitzar la seva forma d'interacció simple per aconseguir que, d'un col·lectiu d'agents reactius simples, n'emergeixi un comportament global complex i robust. El seu principal avantatge és la seva adaptabilitat a entorns que canvien dinàmicament amb una gran rapidesa. Per altra banda, quan s'usen per resoldre algun tipus de problema complex, aquest tipus de col·lectius no solen trobar la solució òptima, però aquest és un camp encara en estudi i amb moltes vies obertes (incrementar la component deliberativa del sistema en una arquitectura híbrida, estudi d'heurístiques en la presa de decisions locals, etc.).
- Agents deliberatius: el seu sistema de raonament es basa en la manipulació simbòlica, sovint estan orientats a un objectiu (*goal directed*) i un planificador s'encarrega de fer el raonament sobre l'estat del món (en funció dels coneixements i representació interna) per decidir les accions a executar. A diferència dels reactius, la seva principal limitació és la incapacitat per treballar en entorns dinàmicament variants (això es fa especialment evident en la robòtica mòbil). Però com en el cas anterior arquitectures híbrides permeten reduir aquesta incapacitat, a més d'altres tècniques com la descomposició en "subagents" que s'encarreguen de tasques específiques.
- Agents socials: tot i que els podríem incloure dins dels deliberatius. Aquí es vol destacar que, dins del model simbòlic que permet fer el raonament intencional, a més, aquest tipus d'agents poden mantenir models d'un o més agents del mateix entorn per raonar sobre ells i fer-ne raonaments. A més existeix una comunicació explícita i intencional amb ells per mitjà d'un ACL (*Agent Communications Language*).

2.5.3 Agents BDI i programació orientada a agents

En el cas dels agents deliberatius, un dels enfocaments tradicionals per a la seva anàlisi i implementació ha estat modelar formalment les components internes de l'agent. En aquest cas es suposa que l'agent utilitza internament una

representació simbòlica de l'entorn per fer els seus raonaments. Normalment aquesta representació és en forma de fórmules lògiques, i les accions i decisions de l'agent seran resultat de les operacions que faci amb elles (utilitzant mecanismes com la deducció lògica, demostració de teoremes, etc). Aquesta formalització ha portat al desenvolupament de diverses arquitectures formals d'agents [76], dintre de les quals trobem les BDI (*Belief Desire Intention*). És necessari introduir, molt breument, aquest tipus d'agents perquè aquest és un dels enfocaments que han generat més recerca i ha derivat en altres camps d'estudi com els llenguatges orientats a agent (*Agent-Oriented Programming Languages*).

Els agents BDI tenen certes '"actituds mentals" que representen els coneixements (*beliefs*) que tenen sobre ells mateixos, els altres agents i l'entorn en que conviuen tots ells, més d'una sèrie de desitjos o *desires* (com, per exemple, els objectius a llarg termini que volen assolir), és a dir, la seva motivació, i les intencions (*intentions*) que reflecteixen els plans immediats (quines accions a curt termini ha previst fer l'agent), és a dir l'estat deliberatiu de l'agent. Així l'arquitectura es basa en el raonament pràctic per anar decidint en cada moment les accions que durà a terme l'agent. Aquest raonament pràctic inclou dos subprocessos principals: el deliberatiu amb el qual l'agent decideix quins objectius intenta assolir en cada moment, i el *means-end reasoning* amb el qual l'agent troba un pla (accions immediates) per assolir aquests objectius.

Les implementacions que s'han fet, basades en aquests *frameworks* teòrics, permeten la construcció d'agents a partir de la seva descripció formal. D'aquí la necessitat d'eines de programació que facilitin aquesta implementació. I és en aquest àmbit que Shoham [72] va introduir el concepte de *Agent Oriented Programming (AOP)*, que dóna la visió d'un sistema composat per un conjunt d'agents (és una evolució natural del paradigma orientat a objecte). La idea és que aquests agents es poden descriure (de forma similar al mecanisme d'abstracció que utilitzem els humans per descriure sistemes complexos) en termes del seu estat mental (els *beliefs, desires* i *intentions* que hem vist abans). Agent0 és el llenguatge que va definir Shoham per la programació d'agents intel·ligents amb una formulació pseudo-BDI (els agents creats ja tenen capacitats deliberatives i de raonament intrínseques). Un Agent0 es defineix amb uns conjunts inicials d'habilitats (o capacitats) , *beliefs*, *commitments i commitment rules*. En el moment de la seva inicialització, l'estat mental de l'agent es composa dels *beliefs* i *commitments* inicials (aquests darrers, encara que s'instancien al començar l'execució de l'agent, potser no seran aplicables fins un cert instant en el futur). Les habilitats són les accions que l'agent és capaç de dur a terme, i són fixes durant el seu temps de vida. Les *Commitment rules* determinen les accions que durà a terme l'agent i els canvis al seu estat mental davant de totes les situacions possibles.

2.5.4 Cap a un sistema multiagent

Tot i que hi ha situacions en que un agent opera tot sol en un entorn (ja sigui computacional, físic o una mica de les dues coses), aquesta situació és cada cop menys freqüent i els agents han d'interactuar amb altres agents. Encara més, a vegades el nombre d'agents és tan nombrós que cal tractar-lo com un col·lectiu, una societat d'agents. Així doncs el concepte de sistema multiagent (MAS) fa referència a l'anàlisi i construcció de sistemes formats per més d'un agent artificial. A partir d'aquesta definició simple ens podríem preguntar quin tipus d'agents dels que hem vist abans formarien aquest tipus de sistemes. A principis dels 90s els investigadors de les escoles reactiva i deliberativa tenien visions diferents en aquest aspecte (consideraven que els MAS estarien formats per agents reactius o deliberatius íntegrament), però els darrers anys és comú trobar MAS amb agents purament reactius, deliberatius i arquitectures híbrides[68]. Arribats a aquest punt podem fer un incís per situar els MAS dins de la DAI (*distributed artificial intelligence*). En la majoria de casos el problema que es vol tractar es composa d'elements distribuïts (potser geogràficament), de caràcter heterogeni (naturalesa diversa), i informació fragmentada i també de diversos tipus. Les 4 principals tècniques per tractar amb problemes d'aquestes dimensions i complexitat solen ser la modularitat, distribució abstracció i intel·ligència. La DAI combina aquestes 4 característiques [37] i dues grans àrees de recerca dins de la DAI s'encarreguen de tractar aquest tipus de sistemes: DPS (*distributed problem solving*) i els MAS. Resumint, DPS és una aproximació *top down* que divideix un problema d'intel·ligència artificial en subproblemes que es poden resoldre de forma distribuïda. En el cas dels MAS l'aproximació és *bottom up* i els problemes es resolen mitjançant la combinació d'entitats que són bones resolent problemes menors però que serien incapaços de resoldre el problema global[68]. D'aquí podem concloure que els MAS tindran les següents característiques :

- Cada agent té una visió limitada del problema. Això vol dir que els agents no tenen prou informació o capacitat de resoldre el problema per sí mateixos.
- No hi ha un sistema de control centralitzat (presa de decisions locals).
- La informació també està descentralitzada.
- És un model de computació asíncron.

El creixent interès en els MAS es deu a la solució robusta i fiable que proporcionen a molts problemes, i a la capacitat que hem vist de treballar amb dades i control distribuït. A més, aquestes característiques possibiliten que es pugui habilitar la cooperació entre sistemes que ja tenim en funcionament (*legacy systems*) i que, fins ara, treballaven de forma no coordinada.

Tot i això, també presenten algunes dificultats o reptes a superar:

- Com poder, d'una manera formal, descriure, dividir i assignar tasques a un grup d'agents intel·ligents?
- Quin serà el model de comunicacions? Protocols? Llenguatges? Què i quan cal comunicar?
- Com coordinar la presa de decisions locals per tal que no afectin negativament al comportament global?
- Quin ha de ser el model que permeti als agents raonar, representar, planificar les accions i modelar el coneixement?
- Com dissenyar tecnologies i metodologies que ens permetin implementar MAS?

2.5.5 Coordinació mitjançant les comunicacions

El agents que formen part d'un MAS , i que, per tant, comparteixen un entorn, utilitzen les comunicacions per interactuar els uns amb els altres. Aquesta comunicació és necessària si es vol que el col·lectiu es coordini per assolir els objectius del sistema . Aquí cal fer notar que els objectius que es volen assolir a nivell global potser no són explícitament coneguts per cada agent individual. En qualsevol cas, el concepte clau que cal aconseguir, és la coordinació dels comportaments i accions del agents perquè el sistema dugui a terme la tasca pel quan ha estat dissenyat.

Tot i que en aquest capítol ens centrarem en les comunicacions explícites, els agents no necessàriament han d'enviar missatges de forma directa (encara que s'usin intermediaris) i intencionada a altres agents per comunicar-se. De fet, el mètode més simple per comunicar-se és a través de la manipulació de l'entorn comú (*stigmergy*). Inspirats per sistemes biològics simples (per exemple els eixams d'abelles o les formigues), els investigadors en robòtica col·lectiva han estudiat els nivells de cooperació que es poden aconseguir amb aquest tipus de comunicació [1][42]. Aquests estudis han mostrat que és possible aconseguir un comportament cooperatiu només manipulant l'entorn, però que la qualitat de la coordinació millora amb una comunicació més directa.

Pel que fa a la cooperació, aquesta no és més que un tipus de coordinació duta a terme per agents no antagònics, és a dir, que no competeixen per l'objectiu o, simplement, ignoren el comportament dels altres. D'aquí que alguns autors[37] classifiquin els mecanismes de cooperació de la forma mostrada a la Figura 3, on també es mostra que la coordinació es pot aconseguir mitjançant agents competitius (antagònics) capaços de negociar entre ells.

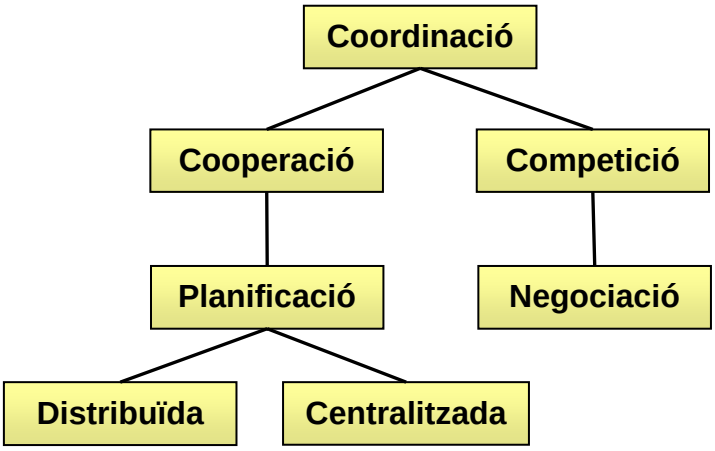


Figura 3. Classificació dels tipus de coordinació entre els agents [37]

Així doncs, si tenen la capacitat de comunicar-se explícitament, els agents intercanvien missatges de forma estructurada, mitjançant la utilització dels protocols de comunicacions. I la interpretació d'aquests missatges és el que dóna sentit a aquesta "conversa" entre ells.

2.6 Llenguatges de comunicacions d'agents

Un cop introduïts els conceptes d'agent i els MAS, assumint que els agents als quals fem referència poden enviar i rebre missatges a través d'alguna xarxa de comunicacions, anem a veure algunes característiques que haurien de tenir aquests llenguatges de comunicacions entre agents (ACL).

Com que els agents han de ser capaços d'interpretar i generar les comunicacions, l'estudi formal d'aquests missatges inclou 3 aspectes bàsics: l'estructura dels símbols (sintaxi), el significat d'aquests (semàntica) i com s'interpreten aquests símbols (quines accions, raonaments interns, etc., duran a terme els agents com a resultat de la interpretació d'un missatge). Per alguns autors el concepte "significat" d'un missatge és una combinació d'aquests 2 darrers aspectes [37]. Altres autors, durant els anys 90, ja es van platejar alguns requeriments que haurien de complir aquests missatges, així a [49] es proposen aquests:

- Forma: un bon llenguatge de comunicacions entre agents hauria de ser declaratiu, sintàcticament simple i intel·ligible pels humans (aquest darrer aspecte no hauria de ser un requisit en sistemes amb restriccions d'optimització importants, com per exemple en alguns sistemes agents físics). També hauria de ser concís i, a la vegada, fàcil de generar i tractar sintàcticament (*parsing*). Per transmetre una ordre d'un agent a un altre, aquesta haurà de passar, convertida en un flux de bits, pels diferents mecanismes de transport subjacents. D'aquí que el llenguatge hauria de ser lineal o fàcilment convertible a lineal. Finalment, degut a que serà

integrat en una àmplia varietat de sistemes, la seva sintaxi haurà de ser extensible.

- Contingut: hauria d'estar format per capes (*layered*) de forma que pugui acoblar-se bé amb altres sistemes. En particular caldria fer una distinció entre el llenguatge de comunicacions, el qual expressa actes comunicatius, i el llenguatge de contingut, que expressa fets sobre el domini. Aquesta estratificació (*layering*) facilita la integració satisfactòria del llenguatge a les aplicacions tot proveint un marc de treball conceptual per entendre el llenguatge.
- El llenguatge ens hauria de proveir d'un conjunt ben definit d'actes comunicatius bàsics (que podríem anomenar "primitives"). Encara que aquest conjunt pot ser extensible, una base de primitives que abasti la majoria de les nostres intuïcions sobre què constitueix un acte de comunicació, independentment de l'aplicació, assegurarà la possible reutilització del llenguatge per una gran varietat de sistemes. Es pot escollir comprometre's amb un llenguatge de contingut específic, això permetria un conjunt de primitives més reduït perquè seria possible transportar més informació a nivell (capa) del llenguatge de contingut, amb el desavantatge que totes les aplicacions haurien d'usar el mateix llenguatge de contingut, i aquesta és una restricció molt forta.
- Semàntica: la semàntica és una característica que sovint s'ha descuidat durant el disseny dels llenguatges de comunicacions. Aquest descuit és un resultat indirecte de l'obscuritat que envolta el propòsit i les prestacions desitjades dels llenguatges de comunicacions. La semàntica d'un llenguatge de comunicacions hauria de tenir una base teòrica i no ser ambigua. Hauria d'exhibir una forma canònica (semblances en significat portarien a semblances en la representació). Finalment la descripció semàntica hauria de proveir d'un model de comunicacions, que seria útil pel modelat de funcionament i una possible verificació funcional, entre d'altres coses.
- Implementació: hauria de ser eficient, tant per velocitat com per ús d'ample de banda. S'hauria d'adaptar bé a la tecnologia software existent. Interfície fàcil d'usar, els detalls de les capes de xarxa subjacents a les primitives haurien de quedar ocults a l'usuari. Hauria de ser possible fer-ne implementacions parcials degut als agents simples que potser en tindrien prou amb un petit subconjunt de primitives.
- Entorn: aquest serà altament distribuït, heterogeni i extremadament dinàmic.
- Fiabilitat: ha de suportar comunicacions fiables i segures. Ha de preveure intercanvi segur de missatges entre dos agents, autentificació. Hauria de suportar mecanismes per identificar i assenyalar errors i advertències.

2.6.1 Converses en ACL: protocols d'interacció

Mitjançant els ACLs els agents poden intercanviar informació i coneixements entre ells. Tot i que aquesta informació pot estar formada per missatges simples, sense semàntica associada als missatges, les capacitats deliberatives i el caràcter social dels agents fan que es puguin intercanviar alguns tipus d'informació més complexes com plans, negociacions, estratègies a llarg termini, etc. Per aconseguir aquest intercanvi d'informació complexa mitjançant les comunicacions, dos o més agents hauran d'acceptar el mateix patró d'intercanvi de missatges que podem anomenar "conversa" (de fet, una conversa és un protocol de coordinació o interacció). Hi ha diverses qüestions que centren l'atenció en l'ús i implementació de protocols d'interacció. En primer lloc l'especificació d'aquestes converses (utilitzat autòmats d'estats finits, notacions més complexes amb restriccions semàntiques i dependències entre missatges, etc.) En qualsevol cas cal un formalisme amb suport a l'anàlisi i la verificació. Altres qüestions d'interès podrien ser com compartir converses (com saber quin tipus de protocols suporta un agent) o bé l'agregació, això és, com combinar protocols estàndard (de forma similar a una API) per descriure un comportament particular. Tota aquesta complexitat d'interacció, semàntica, dependències entre processos, manipulació de coneixements (els ACL més estàndard com KQML o FIPA-ACL manipulen accions, proposicions i regles) fa que es puguin veure els ACLs com una capa de comunicacions que està per sobre de mecanismes més simples com RMI o RPC[43].

2.6.2 Teoria dels actes comunicatius de la parla

Abans de veure quins són ACL més utilitzats, donem un cop d'ull a la teoria subjacent que permet que es puguin utilitzar, de forma estàndard, en dominis d'aplicació de naturalesa ben diferent (accions dels agents específiques, vocabulari associat al domini, restriccions semàntiques i pragmàtica en funció de cada aplicació). La base d'aquesta teoria la trobem al treball del filòsof John Austin [1][6]. Fins aquell moment , l'estudi dels enunciats utilitzats en les comunicacions es centrava en el valor lògic d'aquests enunciats. És a dir, tenint en compte que la majoria d'enunciats es construeixen a partir de fets, els enunciats tindran valor de veritat o fals en funció d'aquests fets. Austin va anar més enllà, fent notar que aquest tipus de frases només són una petita part del total, i es va centrar en un conjunt d'elocucions que no es poden avaluar com a veritat o fals (les anomenades elocucions performatives) . Aquest grup d'elocucions, a més, poden, en certa forma, alterar l'estat del món. Aquestes accions com a resultat de les elocucions donen nom a la teoria d'actes comunicatius o de la parla (*speech act theory*). Així en un acte de parla (una elocució) es distingeixen 3 activitats:

- Acte "locutori": és el fet de pronunciar els sons (en la parla) , emetre la frase. Per exemple "tinc fred".

- Acte "ilocutori": la intenció o significat que té per l'emissor del missatge. Seguint l'exemple anterior, el significat podria ser "vull que algú tanqui la porta".
- Acte "perlocutori": l'acció o canvis que es produiran com a resultat de l'acte de parla. En aquest cas el tancament de la porta.

Fixem-nos que, en el cas del llenguatge natural humà, hi ha una ambigüitat que s'evitarà en els ACL. Tornant a l'exemple anterior, la frase que s'ha pronunciat potser no tenia la intenció de provocar el tancament de la porta, simplement podia ser una afirmació. Un altre aspecte important és que, per causes diverses, els resultats de l'acte de parla no sempre seran els esperats, és a dir, l'emissor no té la certesa que es produeixi l'acció que potser està sol·licitant.

Aquesta distinció va permetre a Austin fer una classificació dels verbs del llenguatge humà en funció de l'efecte que produeixen (peticions,sol·licituds, informació, etc.). Searle va continuar el treball d'Austin fent una classificació més genèrica dels verbs i introduint el concepte de "força ilocutòria"[69]. Aquest concepte s'usarà en els ACL per "encapsular" els missatges dels agents dintre de contenidors que segueixen aquesta classificació. Així la intenció de l'agent emissor del missatge queda definida encara que el contingut del llenguatge després sigui contradictori o no es pugui entendre.

2.6.3 KQML

El *Knowledge Query and Manipulation Language* és un ACL d'alt nivell desenvolupat a principis dels anys 90 i es basa en la *speech act theory*. Va sorgir del treball del consorci KSE (*Knowledge Sharing Effort*), el qual centrava inicialment els seus esforços en la interoperabilitat de sistemes basats en coneixement (de fet, inicialment la paraula "agent" ni tan sols formava part del vocabulari de KSE). No és fins a 3 o 4 anys després dels seus inicis que KQML és rescatat com a llenguatge per a les comunicacions entre agents. La sintaxi de KQML és de tipus llista de parèntesi balancejats, similar a Lisp. A les especificacions inicials el primer element de la llista era una acció i la resta els seus arguments en forma de parells clau/valor (les primeres implementacions es van fer en *Common Lisp*). KQML és un llenguatge de comunicacions d'alt nivell orientat a missatges, i protocol per intercanviar d'informació independent de la sintaxi del context i la ontologia aplicable. És independent del mecanisme de transport (TCP/IP,SMTP,...), independent del llenguatge de context (KIF,SQL, PROLOG,...), i independent de la ontologia assumida pel context. Conceptualment podem identificar 3 capes en un missatge KQML [43]:

- Contingut: la capa de contingut suporta el contingut missatge actual en el llenguatge propi de l'aplicació. KQML pot transportar qualsevol llenguatge de representació, com els llenguatges expressats com a cadenes ASCII i també aquells expressats en notació binària. Les implementacions de KQML ignoren la part de contingut del missatge, excepte per determinar on acaba.

- Comunicacions: la capa de comunicacions codifica un conjunt de característiques al missatge les quals descriuen els paràmetres de comunicació a baix nivell, tals com identitat de l'emissor i receptor.
- Missatge: codifica un missatge que una aplicació voldria transmetre a una altra, és el nucli del KQML. Aquesta capa determina els tipus d'interaccions que podem tenir amb un agent que parli KQML. La funció de la capa de missatge és identificar la performativa (concepte extret de la *speech act theory*) que l'emissor adjunta al contingut. Recordem que, segons la classificació que es fa a la *speech act theory*, aquesta performativa indica la intencionalitat del missatge (una asserció, una pregunta, una comanda o alguna altra d'un conjunt de performatives conegudes). A més, com que el contingut del missatge és opac a KQML, la capa de missatge també inclou característiques opcionals que descriuen el llenguatge del contingut, la ontologia que assumeix, i diversos tipus de descripció del context.

Així doncs, KQML és a la vegada un format de missatges i un conjunt de protocols que permeten la identificació, connexió i intercanvi de coneixements (informació) entre agents (generalitzant, entre aplicacions). Quan un agent utilitza KQML, transmet missatges escrits en el llenguatge que vulgui, embolcallats dins de missatges KQML. Les implementacions de KQML ignoren la part de contingut, excepte per reconèixer on comença i on acaba el missatge. Per exemple, a la Figura 4 es mostra el missatge amb el qual l'agent 1 demana l'agent 2 per un aeroport concret. En aquest cas, la performativa és "ask-all", així la intencionalitat del missatge és sol·licitar totes les respostes possibles al missatge. El missatge és "geoloc(lax,[Long,Lat])" i està escrit en PROLOG. Com que el context és una aplicació d'aeroports i localitzacions geogràfiques, els dos agents utilitzaran la ontologia "geo-model3".

```
(ask-all
:sender agent1
:receiver agent2
:content "geoloc(lax,[Long,Lat])"
:ontology geo-model3
:language standard_prolog)
```

Figura 4. Exemple de missatge KQML [1]

Les performatives o actes de parla de KQML es poden dividir en 7 categories [49]:

- Consultes bàsiques (evaluate, ask-one, ask-all, …)
- Consultes multi resposta (stream-in, stream-all, …)

- Respostes (reply, sorry…)
- Informatives (tell, achieve, cancel, …)
- Generadores (standby, ready, next, …)
- Que poden definir capacitats (advertise, subscribe, monitor, …)
- De xarxa (register, unregister, forward, broadcast, …)

2.6.4 FIPA

La *Foundation for Intelligent Physical Agents* [28] (FIPA) es va crear el 1996 (amb seu a Suïssa) com a associació, sense ànim de lucre, d'empreses i organitzacions amb la intenció de produir especificacions estàndard per agents i sistemes multiagent heterogenis que poden interactuar entre ells. El juny del 2005 la organització suïssa es va dissoldre i FIPA va ser oficialment acceptada per l'IEEE, passant a ser el seu 11è comitè o grup de treball. La intenció és que els estàndards i sistemes basats en agents s'integrin en el camp més ampli del desenvolupament de software ja que han d'integrar-se i treballar amb sistemes no basats en agents. L'any 2002 passen a ser estàndard 25 d'aquestes especificacions. Resumint, algunes d'elles són [26]:

- Arquitectura abstracta: després de les primeres especificacions el 1997 i el 1998 va quedar clar que seria útil donar suport a variacions dels missatges estàndard. Variacions pel que fa al seu transport, representació (binari, XML, etc.) i atributs opcionals (encriptació, autentificació, etc.). També va quedar clar que, per la seva implantació comercial calia entendre i utilitzar entorns software existents (missatgeria, serveis de seguretat i directori, plataformes distribuïdes, connectivitat intermitent, etc.). En comptes d'anar revisant i incorporant de forma incremental aquestes característiques es va optar per crear una arquitectura abstracta que pogués acomodar la majoria de mecanismes usats (serveis de directoris, missatgeria etc). A més, per aconseguir la interoperabilitat i la reutilització entre sistemes diversos cal identificar els elements claus d'aquestes arquitectures, les característiques comunes. I aquesta sèrie d'elements claus i comuns entre arquitectures permeten definir un model d'arquitectura abstracta a partir del qual fer implementacions concretes de sistemes que compleixin les especificacions. Sense entrar en detalls, l'arquitectura especifica el transport de missatges, directoris d'agents i serveis i l'ACL. A més, s'especifiquen algun elements opcionals que poden ser obligatoris en algunes de les implementacions.
- Llenguatge FIPA-SL (*Semantic Language*): aquest llenguatge, i la seva semàntica associada, es proposen com a llenguatge de contingut de FIPA-ACL. Tot i que, tal i com veurem, FIPA-ACL no obliga a utilitzar un llenguatge de contingut concret (en aquest sentit funciona igual que KQML).

- Protocols d'interacció: l'arquitectura abstracta especifica un conjunt d'objectes abstractes que permeten la representació explícita d'una "conversa", és a dir, un conjunt de missatges entre interlocutors que estan lògicament relacionats per algun patró d'interacció. Les especificacions descriuen els següents protocols: Query, Request, Contract Net, Iterated Contract Net, Brokering, Recruiting, Subscribe i Propose. Per exemple, a la Figura 5 es mostra el protocol que permet a un agent demanar-li a un altre que realitzi alguna acció en el moment en que una precondició donada es compleixi .
- FIPA-ACL: els missatges de FIPA-ACL es basen, de la mateixa manera que hem vist en el cas de KQML, en la *speech act theory* per embolcallar el contingut dels missatges, contingut que depèn del domini d'aplicació, dins d'una performativa que reflexa la intencionalitat d'aquest acte comunicatiu. De fet, segons FIPA, el camp "performative" és l'únic obligatori dels que formen l'estructura d'un missatge de l'ACL, encara que es suposa que en la majoria de casos també el formen els camps "sender", "receiver" i "content". En qualsevol cas, si no pot entendre o rebre correctament alguna part del missatge, l'agent podrà respondre amb un "not-understood". La codificació i l'ordre dels paràmetres depèn del tipus d'implementació (FIPA té especificacions per codificacions en XML, text similar a KQML i binari). A més, es poden afegir paràmetres definits per l'usuari, no definits per FIPA, hauran de començar amb el prefix "X-". De totes formes, independentment de la codificació, un missatge inclou els camps que es mostren a la Figura 6 .

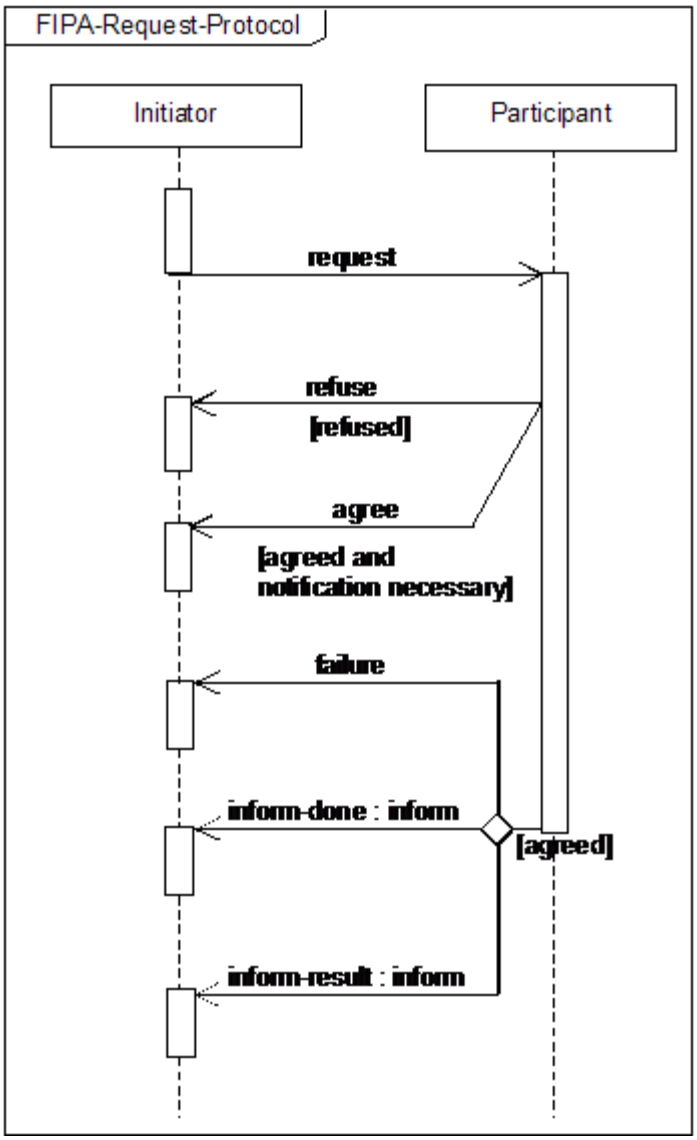


Figura 5. Exemple de protocol d'interacció: FIPA-REQUEST

A la taula s'observa, per exemple, el camp "ontology". La utilització d'una ontologia comú entre els agents participants a la conversa els permet donar significat als símbols, és a dir, ens proporciona una semàntica del llenguatge. També podem veure alguns camps relacionats amb el control de la conversa. Resumint-ho el camp "protocol" ens permet especificar quin dels protocols d'interacció (veure el punt anterior) ha generat el missatge. Altres camps permetran assignar identificadors a converses concretes, especificar quina resposta s'espera, en quin moment etc.

Paràmetre	Categoria del paràmetre
performative	Tipus d'acte comunicatiu
sender	Participant en la comunicació
receiver	Participant en la comunicació

reply-to	Participant en la comunicació
content	Contingut del missatge
language	Descripció del contingut
encoding	Descripció del contingut
ontology	Descripció del contingut
protocol	Control de la conversa
conversation-id	Control de la conversa
reply-with	Control de la conversa
in-reply-to	Control de la conversa
reply-by	Control de la conversa

Figura 6. Paràmetres d'un missatge de FIPA-ACL

2.7 Sistemes multirobot

La descripció més general que es pot donar d'aquest tipus de sistemes és la d'un conjunt de robots que operen en un mateix entorn. Aquesta generalització a l'hora de descriure els MRS (*Multi-Robot Systems*) ens permet fer algunes consideracions importants. En primer lloc, els robots poden ser des de simples sensors amb mínimes capacitats d'adquisició i procés de dades fins a agents intel·ligents amb capacitat de deliberació i interacció amb els humans. En segon lloc, les interaccions dels sistemes robòtics que operen dins de l'entorn poden ser d'una gran complexitat degut a que es requereix un mecanisme de coordinació entre ells. Per tant, podem tenir diversos tipus de coordinació entre els robots, i això dóna lloc a diversos problemes específics que cal resoldre en funció del tipus de mecanisme de coordinació utilitzat (assignació de tasques, comunicacions, mesures d'eficiència, tipus de sistema de control , etc). La tercera consideració que podem fer a partir d'aquesta descripció genèrica fa referència a l'entorn d'aplicació. Tot i que en alguns dominis és relativament controlable, com per exemple en aplicacions industrials (es disposa de mapa complet, els elements de l'entorn es mouen de forma previsible, es poden fer acotacions sobre possibles aspectes crítics), la majoria de MRS estaran situats en un entorn altament dinàmic i imprevisible.

Més enllà de la descripció del concepte, una altra qüestió de debat són les motivacions o justificació per al seu ús, és a dir, quan (i en quines aplicacions) és preferible utilitzar un MRS en comptes d'un SRS (*Single Robot System*). Les principals raons són [25][40]:

- Major eficiència, flexibilitat i robustesa aprofitant la redundància i el paral·lelisme inherents en aquest tipus de sistemes (modularitat). En aquest cas els MRS tenen l'avantatge de disposar de dades de sensors

distribuïts (*local sensing*) a més de robots amb habilitats heterogènies que es poden combinar per aconseguir comportaments emergents a nivell de sistema. Quant a la redundància, fins i tot en el cas dels MRS homogenis tenim avantatges en aquest aspecte. La més evident és, per exemple, que un conjunt de robots iguals poden moure objectes més pesats, o aconseguir moviments més precisos i complexes a l'hora de desplaçar-los.

- Reducció de costos. En moltes aplicacions la utilització de múltiples robots barats redueix els costos respecte la utilització d'un sol robot sofisticat i car. La utilització de robots heterogenis també ajuda a la reducció de costos.

Així i tot, hi ha alguns desavantatges potencials i dificultats a l'hora de dissenyar aquest tipus de sistemes. Un MRS pot arribar a ser molt menys eficient que un SRS si no està ben dissenyat. No n'hi ha prou amb agafar un sistema amb un únic robot, vàlid per l'aplicació, i escalar-lo. De fet el disseny de MRS és un paradigma diferent del de SRS [40]. Per tant, ens cal una metodologia i eines de disseny específiques. Tot i que hi ha diversos treballs en aquest sentit, el disseny d'un RMS encara té algunes etapes que s'han de tractar en funció de l'aplicació.

2.7.1 Taxonomies i dimensions dels MRS

La introducció anterior ens mostra, a més, que els MRS estan caracteritzats per un gran nombre de paràmetres o aspectes possibles que s'han de contemplar en temps de disseny i que suposen tot un camp de recerca de forma individual. Alguns d'aquests aspectes crítics són: el control del sistema (centralitzat o distribuït), el mecanisme de coordinació, les comunicacions (ample de banda, distància d'abast, llenguatge, ...), quantitat de robots, assignació de tasques, tipus de robots.

A la literatura trobem diverses classificacions o taxonomies dels MRS en funció d'aquestes característiques, també anomenades "dimensions" Les taxonomies permeten distribuir les diverses solucions existents dins de l'espai multidimensional de MRS que té com a eixos aquestes dimensions, i situar-hi també el nostre domini d'aplicació. Això ens pot servir d'ajuda, per exemple, a l'hora de decidir quina de les solucions existents, entre les més "properes" dintre d'aquest espai multidimensional, s'ajusta més al nostre domini d'aplicació a l'hora de dissenyar el MRS. La taxonomia proposada a [24] permet fer una classificació basada en les següents dimensions:

- Mida del col·lectiu: en aquest cas es proposa distingir els sistemes segons estiguin formats per un sol robot, un parell, múltiples robots (però en un nombre relativament petit respecte l'entorn d'aplicació) i infinits robots (col·lectiu molt nombrós).

- Abast de les comunicacions: des del cas més simple en que els robots no es poden comunicar (però sí de forma indirecta a través de l'entorn), el cas en que les comunicacions directes estan limitades a una certa proximitat, i, finalment, el cas en que tots els robots es poden comunicar entre ells.

- Topologia de les comunicacions: els robots no sempre es poden adreçar entre ells a l'hora de comunicar-se. Depenent de l'arquitectura del sistema, per exemple, hi ha casos en que tots els robots envien els seus missatges a un agent concret que els redistribuirà, o bé s'han de comunicar seguint una certa jerarquia, potser s'utilitzen mètodes de pissarra, *broadcast*, etc. Per això aquesta característica de la taxonomia distingeix entre sistemes amb broadcast (on no és possible enviar el missatge només a un robot), sistemes on es poden adreçar a un altre robot per nom o adreça, sistemes on la comunicació és en forma d'arbre (els robots es comuniquen seguint una jerarquia havent-hi, per exemple robots "supervisors" que controlen subgrups de robots), i sistemes on la topologia de comunicacions segueix una estructura de graf (és una generalització de l'arbre, així s'evita que parts del sistema quedin potencialment desconnectades).
- Ample de banda de les comunicacions: el cost de les comunicacions pot ser insignificant, en termes de temps de procés del robot, o bé suposar una sobrecàrrega impedint qualsevol acció del robot mentre s'està comunicant. Aquesta dimensió de la taxonomia distingeix 4 casos possibles. En primer lloc tenim sistemes on les comunicacions són lliures i sense cost (ample de banda infinit). En segon lloc, sistemes on el cost de les comunicacions és del mateix ordre de magnitud que el cost del desplaçament del robot (això passa, per exemple, si es vol utilitzar el moviment per transmetre missatges, com fan les abelles quan troben una font d'aliment). En tercer lloc estan els sistemes amb un alt cost de comunicacions, molt superior al cost de moure's d'un lloc a un altre (això suggereix robots força independents). Finalment el cas sense comunicacions, els robots no es poden detectar els uns als altres. En general, els sistemes amb ample de banda baix (comunicacions costoses) solen ser acceptables en casos en que la raó principal d'ús de múltiples robots és treure profit de la redundància més que aconseguir eficiència.
- Reconfigurabilitat col·lectiva: en aquest cas en referència a la capacitat de reorganització dels robots en l'espai. Equivalent a la taxa de "mobilitat" dels robots respecte als altres. Aquesta dimensió està lligada al rang o abast de les comunicacions. Es distingeix el cas on no hi ha canvis de posicionament (topologia fixada), el cas en que el posicionament és coordinat entre membres que es comuniquen, i el cas en que el posicionament és totalment dinàmic.
- Capacitat de procés de cada unitat: cada element robòtic té un model de computació particular, i seria útil modelar les unitats del col·lectiu amb un model de computació simple (per exemple una màquina d'estats finita). D'aquesta manera es facilitaria la seva verificació formal i acotar els possibles comportaments no previstos. Així i tot, aquest model de computació simple individual no impedeix que les capacitats del sistema sencer siguin superiors a les individuals. Aquesta dimensió distingeix des

de les unitats més simples (operacions bàsiques, bàsicament en sistemes tipus xarxa neural, però massa simples per formar part d'un MRS realista), a les màquines de Turing. Entremig es distingeixen els autòmats d'estats finits i els autòmats *push-down*.

- Composició del col·lectiu: fins i tot en col·lectius amb un hardware idèntic, podem tenir software de control o comportaments diferenciats. Aquí podem distingir sistemes amb tots els robots idèntics (a nivell físic i funcional), amb el mateix hardware (però no necessàriament el comportament), i sistemes on el hardware és diferent (això implica, en general, també comportaments diferents).

A [25] es proposa una taxonomia centrada en aspectes de coordinació, per tant, inspirada per la relació amb el camp dels MAS. Aquesta es caracteritza per tenir 2 grups de dimensions: de coordinació i de sistema. El primer caracteritza el tipus de coordinació, i el segon les característiques del sistema que influeixen al desenvolupament del grup. A diferència de l'anterior taxonomia, les dimensions de coordinació s'organitzen de forma jeràrquica tal i com es mostra a la Figura 7.

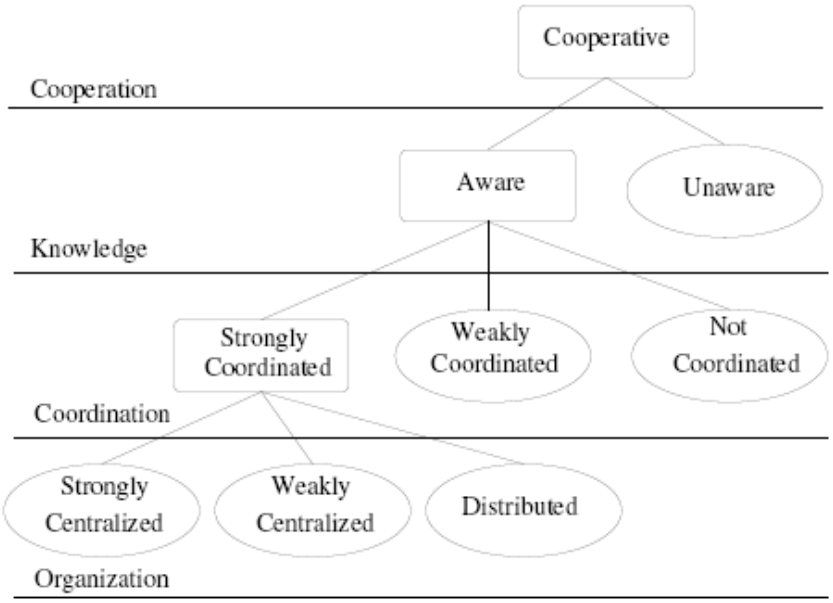


Figura 7. Taxonomia jeràrquica de les dimensions de coordinació. [25]

Aquestes dimensions de coordinació són:

- Nivell de cooperació: distingeix si el sistema és cooperatiu o no. Aquesta taxonomia es centra en els MRS cooperatius, entenent com a cooperatius els sistemes formats per robots que operen plegats per assolir alguna tasca global. Aquesta noció de cooperació és similar a la que hem vist amb els MAS, però amb una diferència respecte aquests: tal i com hem vist a la Figura 3, dins de la coordinació, la cooperació es contraposa o uneix a la

competició. Fins al moment, excepte algun cas remarcable, la competició ha rebut poca atenció en el camp del MRS.

- Nivell de coneixement: aquest segon nivell fa referència al coneixement que té cada robot sobre l'existència d'altres robots amb els quals treballa en equip. Els *aware* en tenen algun tipus de coneixement, mentre que els *unaware* actuen sense tenir coneixement dels altres. Des del punt de vista de l'enginyeria, els MRS cooperatius *unaware* són interessants degut a la seva simplicitat. Cal observar que aquesta noció no és equivalent a les comunicacions. Hi ha MRS *aware* sense comunicacions i a la inversa.
- Nivell de coordinació: el tercer nivell fa referència als mecanismes de coordinació. Aquí es considera que el mecanisme de coordinació de cada robot té en compte les accions executades pels altres robots. Com que els robots poden tenir diverses formes de "tenir en compte" les accions dels altres, aquesta taxonomia es basa en el protocol de coordinació utilitzat pels robots per tal d'interactuar amb els altres i amb l'entorn. Així, es poden classificar els MRS coordinats de manera "forta" (*strong*) o "dèbil" (*weak*) en funció de si el mecanisme de coordinació es basa en un protocol, o no. A diferència dels MAS, les dificultats per utilitzar un protocol de coordinació en algunes aplicacions fan especialment interessants els sistemes dèbilment coordinats.
- Nivell d'organització: aquest nivell fa referència a com s'implementa dins el MRS el sistema de decisions. Aquí es distingeixen les solucions centralitzades de les distribuïdes.

Les dimensions del sistema són les següents (en algun cas coincideixen parcialment amb les que hem vist a la taxonomia anterior):

- Comunicacions: tot i que els mecanismes de comunicacions entre robots poden ser molt diferents, en aquesta taxonomia només es distingeix entre comunicació directa i indirecta (aquesta darrera mitjançant la manipulació de l'entorn o *stigmergy*).
- Composició del grup: aquí es distingeix entre grups de robots homogenis on tots els robots tenen exactament el mateix hardware i software de control, i grups heterogenis on els hardwares o el software de control poden ser diferents.
- Arquitectura del sistema: aquesta característica fa referència a l'arquitectura del sistema sencer, no a la d'un únic agent robòtic. Es fa una distinció entre arquitectures deliberatives i reactives. Es considera que una arquitectura és deliberativa si permet que els robots afrontin els canvis a l'entorn proporcionant una estratègia per reorganitzar els comportaments de tot el grup. Per altra banda, a les arquitectures reactives cada robot reacciona als canvis de l'entorn amb una estratègia local (reorganitzant les seves pròpies tasques) per tal d'assolir l'objectiu que li ha estat assignat.

- Mida del col·lectiu: tot i que hi ha alguns treballs recents sobre col·lectius nombrosos de robots, a la majoria de MRS el nombre de robots no és massa elevat . en aquesta taxonomia, més que un valor quantitatiu sobre el nombre de robots al sistema (com a la taxonomia anterior), es vol distingir els sistemes que, com a opció de disseny, consideren la opció de tractar amb un gran nombre de robots enfront d'aquells que no ho fan.

I, continuant amb les característiques de coordinació, una qüestió que es tracta de forma implícita a tots el mecanismes de coordinació de MRS és "quin robot hauria d'executar cada tasca?". Aquesta qüestió és a l'arrel del problema del MRTA (*multi-robot task allocation*). En els mecanismes de coordinació explícits aquesta es sol tractar directament mitjançant la comunicació intencionada o la negociació entre robots. En la coordinació implícita això es fa de forma indirecta, assignant els robots a tasques produïdes a causa de les interaccions locals i efectes emergents. Els tres eixos següents formen la taxonomia de problemes MRTA proposada a [40]:

- *Single-task robots (ST) vs. multi-task robots (MT):* als ST cada robot només pot executar una tasca a la vegada com a màxim. Als MT alguns robots poden fer diverses tasques a la vegada.
- *Single-robot tasks (SR) vs. multi-robot tasks (MR):* SR vol dir que cada tasca requereix exactament un robot per assolir-la. MR vol dir que algunes tasques poden requerir múltiples robots.
- Instantaneous assignment (IA) vs. time-extended assignment (TA): al IA, la informació disponible sobre els robots, les tasques i l'entorn permet només una assignació instantània de tasques als robots, sense planificar futures associacions. Al TA hi ha més informació, tal com el conjunt de totes les tasques que s'hauran d'assignar, o un model de com s'espera que les tasques arribin durant el temps.

Amb aquesta classificació s'intenta veure com diversos problemes MRTA poden situar-se en el mateix espai de problemes (i així buscar possibles solucions en aquest espai més genèric). Per exemple, els problemes ST-SR-IA es situarien al *Optimal Assignment Problem*, els ST-SR-TA requereixen construir una planificació al llarg del temps per cada robot, i els ST-MR-IA es coneixen com a formació de coalicions.

2.7.2 MRS i el model multiagent

Els dominis d'aplicació per als quals són viables i profitosos els MRS tenen les característiques bàsiques que hem vist al capítol 2.5.4 a l'hora de descriure els sistemes multiagent. A més, si considerem que els MRS es composen de robots autònoms, aquests robots compleixen també les característiques bàsiques d'agent que hem vist al capítol 2.5.1. Així doncs, la visió d'un sistema multirobot format per un conjunt d'agents físics ens permet adoptar el paradigma multiagent com a model computacional en el qual basar-nos a l'hora d'estudiar o implementar els

MRS. El model multiagent ens ofereix abstraccions d'alt nivell mecanismes i eines per representar el coneixement, el model de comunicacions, la coordinació. Molts models, eines i arquitectures formals del camp dels agents es poden aplicar al camp dels MRS.

De totes formes, els MRS presenten algunes característiques i problemes específics causats pel fet d'haver de treballar en un entorn d'aplicació físic, com, per exemple, la informació incomplerta (talls a les comunicacions, sensors de curt abast, obstacles "opacs", ...), incertesa d'aquesta informació (entorn dinàmic amb objectes que es mouen , interferències als senyals, entorn "infinit",...). A més existeixen restriccions hardware evidents, ja sigui pel nivell tecnològic requerit, per restriccions de l'entorn o degudes al seu cost. Per això algunes solucions poc utilitzades en els MAS poden ser la única opció viable en col·lectius robòtics,o donar resultats millors (i a l'inrevés), i algunes característiques àmpliament utilitzades en els MAS sense cap cost crític es converteixen en possibles colls d'ampolla i objecte d'estudi als MRS. Per exemple, als sistemes robòtics les comunicacions directes es basen en dispositius hardware específics, amb rangs de comunicacions i amples de banda possiblement restringits, a més de costosos. Això fa que, a diferència dels MAS, la comunicació indirecta i els sistemes "dèbilment" coordinats siguin objecte de més estudis dins del camp dels MRS. En altres ocasions es fa necessària alguna modificació o ampliació del model MAS a les necessitats dels MRS. Això es pot veure en els diversos treballs que s'han fet per tal de poder incorporar els ACL estàndard a les comunicacions entre robots. Aquest model de comunicacions entre agents, mentre que s'ha adaptat bé a les comunicacions relacionades amb negociació o la transferència d'informació d'alt nivell, té sobrecàrregues significatives als sistemes que requereixen la transferència de dades de baix nivell, o sistemes amb limitacions restrictives de temps i ample de banda. Els sistemes robòtics reals, sovint estan a dins d'aquestes dues categories, per una banda poden requerir transferir vídeo(dades a baix nivell), i per l'altra l'enviament de dades de sensors a alta freqüència, codificats dintre d'un ACL, pot suposar una sobrecàrrega al sistema de comunicacions. Propostes com la introducció de *backchannels [10]*, canals específics per la transmissió de dades a baix nivell, en paral·lel als canals dels ACL, intenten resoldre aquestes limitacions.

2.7.3 Complexitat de les comunicacions als MRS

Com hem vist anteriorment, el mecanisme de coordinació entre els robots pot requerir que aquests hagin d'intercanviar informació de forma directa per assolir els seus objectius. De fet, es pot considerar que la quantitat d'informació que s'ha de compartir determina com de "coordinada" és una tasca (ja que compartir més requereix més recursos). Les restriccions físiques dels dispositius de comunicacions, el tipus d'informació específica que necessiten els robots, juntament amb un entorn dinàmic requereixen analitzar els requeriments de d'aquestes comunicacions i establir les millores en rendiment que poden aportar al sistema. Els investigadors en processament paral·lel saben els efectes

adversos que podem trobar: afegint més elements de procés redueix el temps de càlcul fins un cert punt a partir del qual el temps necessari per la coordinació entre processos supera els avantatges de tenir més processadors. Amb els MRS passa una cosa similar. Si la tasca requereix un gran tractament de coordinació, aleshores el rendiment del sistema es degradarà com és robots hi afegim. I, encara més, pot haver-hi aplicacions en les quals la introducció de comunicacions explícites no proporciona cap millora en el rendiment global del sistema, i, per tant, no cal dedicar-hi recursos.

Per intentar evitar que les comunicacions es converteixin en un coll d'ampolla del sistema seria necessari disposar d'alguna mesura, mètrica o mecanisme d'avaluació del cost de les comunicacions, el seu rendiment o la seva complexitat. Klavins [41] utilitza el concepte de "cost abstracte" (cost d'un event de comunicacions, independentment del medi utilitzat, en termes d'ample de banda, temps de latència, etc.) per calcular la complexitat de coordinació dels algorismes de control dels robots (i així poder investigar la seva escalabilitat). Això ho fa sumant els costos de tots els events de comunicacions a cada pas d'execució de l'algorisme de control (finalment s'agafarà la mitja de tota l'execució de l'algorisme). Amb aquesta definició es fa present que la complexitat del pitjor cas és $O(n^2)$: tots els robots es comuniquen constantment amb els altres, l'ample de banda de les comunicacions creix amb el quadrat del nombre de robots. Per aquesta raó un algorisme o tasca de complexitat $O(n^2)$ no es considera escalable. Per altra banda, en un algorisme O(n) , la complexitat només creix linealment amb el nombre de robots (afegint un nou node de comunicacions no hi ha un decreixement del rendiment).

Una altra pregunta que hauríem de respondre és: donada una tasca, quina és la mínima quantitat de comunicacions requerida per assolir-la per qualsevol nombre de robots?. Per exemple, a l'esquema de comunicacions DMC (*Distance Modulated Communication*) [41] un robot només necessita un càlcul acurat de la posició dels robots propers (per exemple per evitar col·lisions), mentre que el càlcul pot ser menys acurat per robots llunyans. En aquest cas l'esquema conté només 2 robots que es comuniquen a una freqüència proporcional a la distància entre ells (complexitat O(n log n) o $O(n^{1.5})$) depenent de la dinàmica dels robots. Un altre esquema , MCS (*Wandering Communication Scheme*) [41], defineix un protocol en el qual un petit nombre (constant) de robots "deambuladors" tenen permís per moure's i comunicar-se, mentre els altres romanen immòbils. Un deambulador pot transferir el seu dret a moure's i la seva informació del món a un robot immòbil en un *stream* curt de comunicacions. S'ha comprovat que WCS té complexitat O(n) o O(1) depenent de les assumpcions d'algunes decisions d'alt nivell dels robots.

Amb l'objectiu de mesurar l'efecte de les comunicacions als MRS, Balch i Arkin [7] utilitzen una mètrica del rendiment perquè aquest sigui objectivament mesurable. Aquesta mètrica és un dels punts clau a la metodologia de disseny que proposen, la qual es basa en la maximització (o minimització) d'aquesta mètrica en un cicle iteratiu de simulació/implementació. La selecció de la mètrica de rendiment és

important perquè questes mètriques estan sovint en competició (per exemple cost enfront de fiabilitat). Algunes mètriques potencials per MAS robòtics són:

- Cost: construir el sistema per fer una tasca amb el mínim cost. Pot ser apropiat per tasques industrials. Tendeix a minimitzar el nombre de robots usats.
- Temps: assolir la tasca en el mínim temps. Aquesta mètrica ens portarà a una solució que requereix el nombre màxim de robots que poden operar sense interferir-se.
- Energia: en aquest cas cal usar la mínima quantitat d'energia. Útil per exemple en aplicacions a l'espai o sota el mar.
- Fiabilitat: el sistema ha de tenir la major probabilitat de completar la tasca, encara que sigui a expenses del temps o el cost. Per exemple en algunes operacions militars estratègiques.

A més, també poden ser combinació numèrica de diverses mesures. Els primers resultats d'aquest estudi d'impacte de les comunicacions en grups de robots els van fer comparant tres tasques simples i prou genèriques: *forage*, on els robots ronden per l'entorn buscant punts d'atenció (anomenats *attractors*) als quals s'han d'enganxar i dur a una base; *consume ,* semblant al *forage* però en aquest cas cal fer alguna operació sobre l' *attractor* sense dur-lo a una base, i *graze*, on no hi ha *attractors* i cal visitar (o cobrir) la màxima extensió d'entorn possible.

Com que la intenció és veure com d'escalable (quant al nombre de robots) és el sistema, introdueixen una altra eina de mesura basada en el rendiment anomenada *speedup,* que, si P[i,j] és el rendiment per "i" robots i "j" *attractors*, es defineix com es mostra a la Figura 8 :

$$S[i,j] = \frac{P[1, j] / i}{P[i, j]}$$

Figura 8. Speedup del rendiment per "i" robots i "j" attractors [7]

Per exemple, suposem que la mètrica de rendiment és el temps que triguen els robots a assolir la tasca. Fixem-nos que, si dos robots triguen just la meitat en fer una feina que un de sol, aquest valor és 1. És clar que valors superiors a 1 són bons. En qualsevol cas, però, un *speedup* igual a 1 es considera lineal, superlineal si és superior a 1 i sublineal si és menor que 1. També cal destacar que, en alguns casos més robots seran més ràpids a executar la tasca, però encara tindrem un *speedup* sublineal. Els primers resultats experimentals a [7] van mostrar que incorporant comunicacions hi ha una mitja d'un 10% en guany de rendiment. També van observar que, en el cas de la tasca *graze* els resultats són molt similars en tres casos: sense comunicacions, comunicant només l'estat intern

i comunicant informació referent a l'objectiu. Seria un exemple de situació en que no cal incorporar comunicacions explícites, n'hi hauria prou, per exemple, amb la informació proporcionada per alteracions a l'entorn (com un rastre).

MacLennan [46] també es va interessar per l'estudi de les comunicacions mitjançant la "etologia sintètica". Aquesta aproximació es basa en l'estudi del comportament d'organismes simulats que operen en móns sintètics, i va concloure que les comunicacions poden evolucionar en una societat d'agents robòtics simples. Als seus estudis les societats en les quals van evolucionar les comunicacions eren un 84% més eficaces que aquelles on es van suprimir les comunicacions. Un ordre de magnitud en millora de prestacions es va observar quan es va introduir l'aprenentatge.

2.8 Arquitectures de control de robots

El desenvolupament dels sistemes robòtics no es limita a la construcció dels dispositius físics del robot (sensors, motors, ...) i a la implementació del software de control d'aquests dispositius a baix nivell, més les capacitats de procés d'aquesta informació (segmentació d'imatges, reconeixement d'obstacles, ...). Operar en entorns dinàmics requereix una resposta apropiada als canvis en l'entorn dintre de les restriccions temporals apropiades. Ens cal, per tant, un mecanisme d'arbitratge i coordinació entre els diversos comportaments i processos d'informació que pugui tenir el robot. Aquest mecanisme d'arbitratge estarà situat a un nivell superior a aquests mòduls bàsics.

Per exemple, un robot que navegui entre dos punts esquivant obstacles podria planificar la ruta i tenir un comportament bàsic consistent en anar seguint el pla de ruta i, a més, decidir i executar les accions adequades per esquivar els obstacles imprevistos que es presentin. Naturalment aquest segon comportament té unes restriccions temporals més estrictes, per tant, davant d'un obstacle imprevist, la planificació de la ruta cap a l'objectiu (per exemple, com a solució al problema del viatjant de comerç) no pot interrompre el comportament d'evitar col·lisions. A més, aquest arbitratge també hauria d'evitar comportaments contradictoris. En aquest exemple, si es decideix seguir una trajectòria en un sentit, enviant ordres als motors però sense girar les rodes, però el comportament de recerca d'objectius decideix donar la volta al robot per enfocar la càmera, enviant ordres a les rodes per girar-les, tindrem que els dos comportaments envien ordres contradictòries.

Així doncs, les arquitectures de control proporcionen un marc de treball per implementar aquest mecanisme d'arbitratge, de comunicació entre els mòduls de baix nivell dels robots, i d'organització d'aquests (software i hardware) de forma que el sistema compleixi tots els requeriments a baix nivell de l'aplicació a la vegada que el robot es comporta com a una entitat intel·ligent que opera en un entorn dinàmic. Una arquitectura ens proporciona una abstracció a alt nivell del

sistema complert, per sota de la qual es connecten, i es comuniquen entre ells, els diversos mòduls.

2.8.1 Arquitectures Sense-Plan-Act (SPA)

Aquesta primera generació d'arquitectures es basava en el cicle tradicional d'execució dels algorismes (recollir dades, processar-les i treure resultats) per controlar la operativa del robot. D'aquesta manera el control del robot consistia en recollir dades dels sensors, planificar les noves accions a dur a terme en funció d'aquesta informació (calia construir el model del món amb les dades obtingudes), i actuar en conseqüència (d'aquí el nom "*Sense-Plan-Act*"). Aquest és un model deliberatiu que segueix la metodologia tradicional de la intel·ligència artificial consistent en construir un model del món i raonar simbòlicament sobre ell. També es coneixen com a arquitectures jeràrquiques, degut a aquesta distribució en seqüència de les unitats funcionals, tal i com es mostra a la Figura 9.

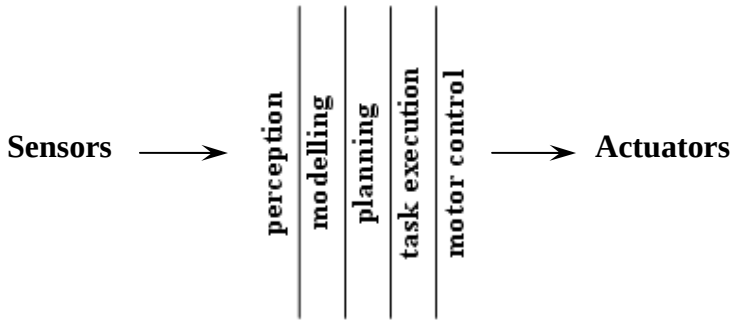


Figura 9. Arquitectura SPA amb jerarquia de mòduls funcionals [13]

Un dels exemples més famosos d'aquestes arquitectures va ser el robot Shakey [57], desenvolupat al "Stanford Research Institute", a principis dels anys 70. La seva arquitectura jeràrquica de control es composava de 5 nivells. El més baix consistia en el vehicle i el hardware de control del robot (controlat per un ordinador remot). Els segon nivell el formaven els programes de baix nivell que accedien a aquest hardware. Aquests programes,anomenats "*Low-Level Actions*" o LLAs, proporcionaven funcions primitives (com per exemple "ROLL" o "TILT") accessibles pels programes escrits en Lisp de més alt nivell. AL tercer nivell de l'arquitectura, les LLAs es combinaven en paquets preprogramats anomenats "*Intermediate-Level-Actions*" (ILAs). Aquests es poden veure com a habilitats instintives del robot ("PUSH", "GO TO", ...). Com que el sensor principal del robot era una càmera, els mòduls de procés d'imatges es limitaven a unes poques rutines de detecció d'objectes i d'orientació del robot. Aquestes rutines estaven repartides entre les LLAs i les ILAs. Al quart nivell, el mecanisme bàsic de planificació (anomenat STRIPS) s'encarregava de construir seqüències d'ILAs per dur a terme tasques específiques. Aquesta seqüència, juntament amb els seus efectes previstos, es podria representar amb una taula anomenada MACROP

(*Macro Operation*). El cinquè nivell era l'executiu, el programa encarregat d'invocar i monitoritzar l'execució de les ILAs especificades en una MACROP.

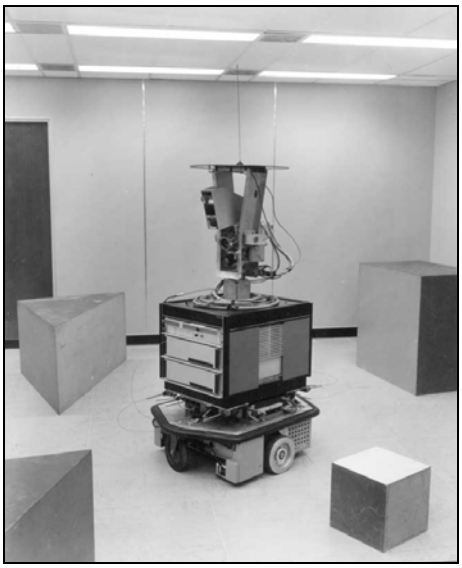


Figura 10. Shakey, exemple d'arquitectura SPA [57]

Shakey era capaç de fer tasques de navegació simples, operant en un entorn real però amb fortes restriccions (superfícies planes, il·luminació intensa i regular, 3 colors a les habitacions, els objectes eren paral·lelepípedes,...). I el seu model del món consistia en un a col·lecció de sentències de càlcul de predicats dins d'una estructura de dades indexada. El problema d'aquest sistema, comú a les arquitectures SPA era la seva incapacitat per operar en entorns del món real, on era capaç de construir un pla d'acció a partir de la seva representació interna del món, però no dintre de les restriccions temporals adients per respondre a temps als imprevistos.

2.8.2 Arquitectures reactives

Vista la incapacitat de les arquitectures SPA enfront les operacions més senzilles al món real, alguns treballs es van centrar en la recerca de mètodes que no es basessin en el raonament a alt nivell, amb una mínima representació interna del món [14]. Es buscava una arquitectura capaç de respondre ràpidament al dinamisme de l'entorn, així les solucions proposades seguien una estructura "*bottom up*" on els mòduls bàsics estaven dedicats a donar resposta ràpidament a les possibles situacions bàsiques en que es poden trobar els robots. El fet que aquests mòduls bàsics es puguin considerar "comportaments" ha fet que, fins fa poc, es considerés l'etiqueta de "robòtica reactiva" sinònim de "robòtica basada en comportaments" (*behavior-based robotics*). Les dues arquitectures més conegudes d'aquest tipus són la *Subsumption Architecture* i la *Schema-based Architecture*

Subsumption Architecture

Brooks publica el treball que dóna a conèixer la *Subsumption Architecture* [13]. el 1986. Aquest considera que els requeriments per un sistema de control robòtic són:

- Múltiples objectius: a vegades contradictoris (per exemple seguir un punt mentre s'eviten obstacles).
- Múltiples sensors: tots els errors tenen una component d'error, dades inconsistents, etc.., el robot ha de fer suposicions sota aquestes condicions.
- Robustesa: El robot ha de ser robust.
- Additivitat: possibilitat d'afegir nous comportaments al sistema.

Les decisions en el disseny de l''arquitectura es van prendre en base als següents 9 principis dogmàtics:

- El comportament complex no ha de ser necessàriament producte d'un sistema de control extremadament complex. Més aviat, un comportament complex ha de ser simplement el reflex d'un entorn complex, hauria de ser un observador qui notés la complexitat d'un organisme, no necessàriament el seu dissenyador.
- Les coses haurien de ser senzilles. Així, al construir un sistema complex, si la interfície entre dos mòduls comença a ser complexa potser caldrà repensar la descomposició del sistema (o el mòdul implicat).
- Volem fer robots de baix cost, que puguin habitar espais on no hi ha humans ni intervenció humana i, a la vegada, fer un treball útil.
- El món humà és tridimensional, no bidimensional, per això el robot ha de tenir un model tridimensional del món.
- Els sistemes de coordinades absolutes són font de grans errors. Per un robot mòbil són més útils els mapes relacionals.
- Els robots es mouran en entorns que no estan formats per poliedres perfectes i simples. Per això no es construirà cap entorn artificial pel robot.
- Les dades de sonar no proporcionen descripcions riques del món, millor reservar-les per interaccions de baix nivell com esquivar obstacles. Millor utilitzar informació visual.
- Per ser robust, el robot ha de seguir operatiu encara que un o més sensors fallin o comencin a fer-ho. Això implica que portarà un sistema de calibratge automàtic, no caldran procediments de calibratge externs.
- Volem fer robots que es preservin, que puguin sobreviure llargs períodes de temps sense assistència humana en entorns dinàmics.

Com a idea principal d'aquesta arquitectura, Brooks proposa la descomposició del sistema en base a comportaments (mòduls orientats a resoldre problemes concrets), trencant així amb la tradicional descomposició en unitats funcionals que feien les arquitectures SPA. És a dir, en comptes de dividir el problema en les tasques internes que ha de fer el robot, proposa dividir el problema en les manifestacions externes (comportaments) que ha de tenir aquest sistema de control (aquesta idea és conseqüència del primer principi dogmàtic que hem vist anteriorment). Per això defineix un conjunt de "nivells de competència" per un robot mòbil autònom. Aquests nivells són especificacions informals dels tipus de comportaments que voldríem per un robot que operi a qualsevol entorn real , com més alt és aquest nivell més específic és el tipus de comportament. A la seva primera implementació aquests nivells de competència eren:

Evitar el contacte amb objectes (tant si es mouen com no).

Desplaçar-se sense cap objectiu específic evitant topar amb els objectes.

"Explorar" el món visualitzant llocs accessibles a la distància, i dirigir-se cap a ells.

Construir un mapa de l'entorn i planificar plans de ruta d'un lloc a l'altre.

Notificar canvis a l'entorn "estàtic".

Raonar sobre el món en termes d'objectes identificables i fer tasques relacionades amb certs objectes.

Formular i executar plans que impliquin canviar l'estat del món d'alguna forma desitjada.

Raonar sobre el comportament dels objectes del món i modificar els plans d'acord amb aquests raonaments.

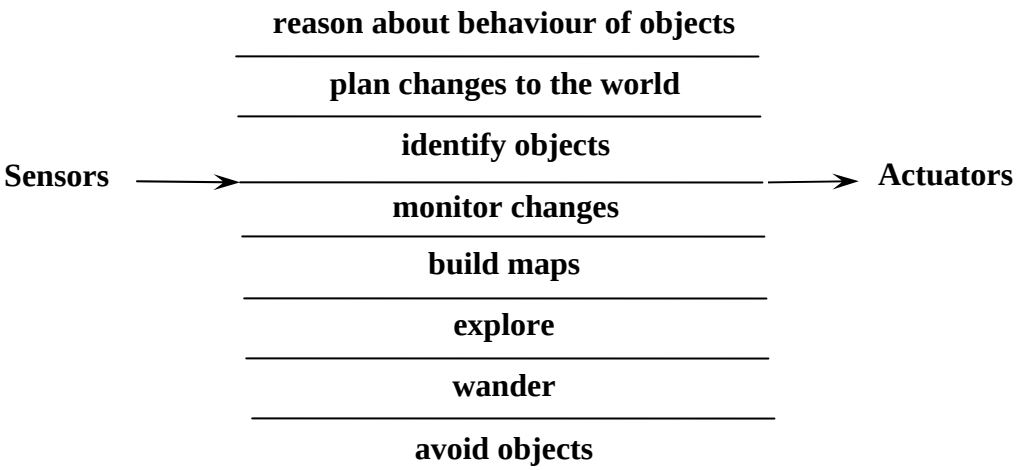


Figura 11. Descomposició del control basada comportaments [13]

Fixem-nos que cada nivell de competència comprèn un subconjunt dels anteriors. Com que cada nivell defineix un conjunt de comportaments, els nivells superiors hi afegeixen restriccions. Aquesta visió vertical de l'arquitectura es mostra a la Figura 11 en contraposició a l'arquitectura jeràrquica de la Figura 9. L'estratificació és la idea clau dels nivells de competència, ja que podem construir capes d'un sistema de control que es corresponguin amb cada nivell, i, simplement, caldrà afegir noves capes per implementar nous comportaments o reemplaçar-ne d'existents.

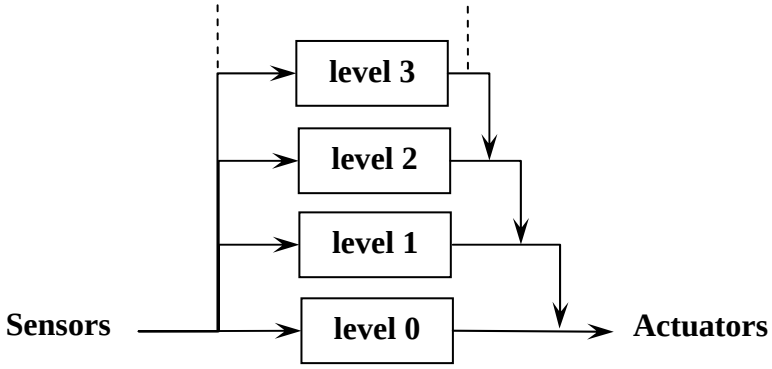


Figura 12. Control estratificat. Es pot partir a qualsevol nivell i les capes inferiors segueixen formant un sistema de control complert i operatiu [13]

La construcció d'un sistema d'aquest tipus es basa en construir un sistema de control complert pel nivell 0, al qual anomenarem "zero". Aquest comportament bàsic es pot depurar i verificar, i no caldrà modificar-lo més. Així tenim una implementació ràpida d'un primer comportament, tot i que simple. A continuació es construeix una altra capa que anomenaríem primer nivell de control i que implementa el comportament 1. Per fer-ho por accedir a la informació de sortida de zero quan li convingui, o bé interferir el comportament d'aquest accedint a les seves interfícies, de forma que pot alterar les entrades del nivell zero, o bé suprimir-ne (substituint-la amb la seva pròpia informació) la sortida cap els actuadors. En qualsevol cas, la capa zero es segueix executant de forma independent sense tenir en compte possibles capes superiors, tot i que aquestes és possible que li suprimeixin les sortides. Aquest procediment es repeteix pels nivells superiors, i aquesta capacitat d'anular o controlar els nivells inferiors quan es vol prendre el control li dóna nom (*subsumption*). L'esquema d'aquesta supressió o alteració del flux de dades de les capes inferiors, juntament amb la introducció de l'asincronisme entre els mòduls que implementen els nivells de comportament es mostra a la Figura 12. Es tracta, doncs, d'un sistema competitiu on, en cada moment, només un comportament pren el control dels motors.

Aquesta arquitectura satisfà els quatre requeriments que hem vist anteriorment:

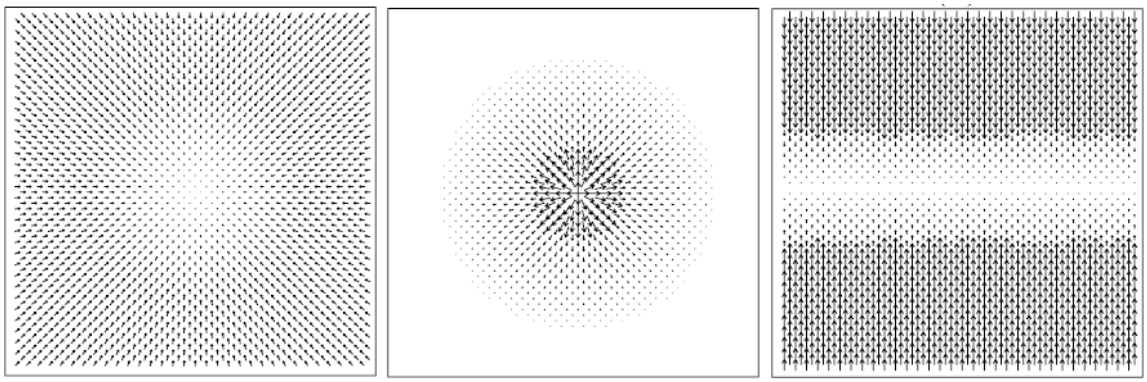
- Múltiples objectius: les capes individuals poden treballar en objectius individuals concurrentment. El mecanisme de supressió controla les accions a dur a terme. L'avantatge és que es deixa pel final quin és l'objectiu que es persegueix.
- Múltiples sensors: aquí no hi ha problema de fusió de sensors, no totes les lectures s'han d'incloure en una estructura de dades centralitzada. Només els valors realment imprescindibles podrien anar cap a una representació central. A més, cada capa agafa les dades dels sensors que li interessen, per tant, una mateixa lectura pot estar sent de forma diferent i a la vegada per cada capa.
- Robustesa: múltiples sensors afegeixen robustesa si són usats intel·ligentment. A més, la *Subsumption Architecture* té una altra font de robustesa. Les capes superiors només poden suprimir les sortides de les inferiors de forma activa, reemplaçant dades, per tant si fallen i no envien sortida, la funcionalitat de les capes inferiors segueix completa.
- Additivitat: la manera òbvia d'aconseguir-ho és afegir capes superiors.

2.8.3 Arquitectures basades en esquemes

Així i tot, el model de capes de la *Subsumption Architecture* presenta alguns desavantatges degut, bàsicament, a dos motius [5]: en primer lloc la competitivitat entre capes fa que en tot moment només un dels comportaments (el més prioritari) controli les sortides dels motors i això li resta flexibilitat en temps d'execució. En segon lloc el fet que els nivells de comportament superiors es basin normalment en els inferiors (als quals afegeixen restriccions) dificulta el disseny perquè el model no és completament modular. Aquests dos inconvenients no els presenta el model basats en esquemes on el comportament és emergent, resultat de la combinació de les sortides dels diferents comportaments individuals. El 1989 Arkin [1] presenta els seus primers resultats experimentals (bàsicament simulacions), juntament amb el model basat en esquemes motor, que té les següents característiques:

- Esquemes (*schema*): són els mòduls bàsics de comportament, asíncrons, Com a concepte d'inspiració biològica (adoptat de la neurobiologia, l'etologia i la psicologia cognitiva)[3], es podria considerar que un esquema és una acció de control automàtica com, per exemple, "escapar", "detectar presa", etc. Les seves sortides poden ser les entrades d'altres esquemes o bé un vector que codifica la resposta comportamental per un estímul donat. Aquesta codificació de la sortida en forma de vector (Arkin l'anomena "vector de velocitat") és una variant de la metodologia dels camps potencials (*potential field*) [35][75]. Els camps potencials s'han utilitzat, bàsicament, en tasques de navegació, on comportaments molt simples es combinen per proporcionar al robot el vector d'acció (*action vector*). La idea és considerar que el robot navega guiant-se pels gradients d'energia

(diferències de potencial) de diverses característiques topològiques de l'entorn. Aquestes característiques topològiques, els camps potencials (*potential fields*), normalment estan definides pel dissenyador i es corresponen amb els comportaments bàsics del robot. D'aquesta manera cada camp potencial proporciona un vector d'acció al robot en un punt del món, format per una orientació i la velocitat en aquest punt. Per exemple, a la Figura 13 s'observa a (a) el camp potencial de "anar cap a l'objectiu, format per vectors que "apunten" a l'objectiu, mentre que a (b) el d'"evitar obstacle estàtic" estarà format per vectors que "repel·leixen" l'obstacle, és a dir, es dirigeixen fora de l'obstacle. En cada moment, el robot només processa la part del camp potencial que està dins el seu rang de sensors. D'aquesta manera l'especificació dels comportaments es fa mitjançant la combinació d'aquests, de forma que un comportament estarà format, normalment, per un esquema "perceptual" i un esquema motor. Comportaments més complexos poden tenir una part perceptual formada per un conjunt d'esquemes (el mateix passa amb la part motor). Els esquemes perceptuals s'encarreguen de processar dades dels sensors, (un esquema perceptual podria ser, per exemple "detectar portes"), mentre que els esquemes motors tenen, com a sortida, efectes sobre els actuadors.



a) Anar cap a l' objectiu. b) Evitar obstacle estàtic c) Mantenir-se al camí

Figura 13. Exemples de camps potencials [4].

- Instàncies d'esquemes: a diferència d'altres models, com el de Brooks, els esquemes són models funcionals que es poden instanciar amb un conjunt de paràmetres inicials. Això vol dir que diverses instàncies poden servir per aprofitar el coneixement i la funcionalitat del mòdul (esquema) a l'hora d'afrontar situacions diverses. Per exemple, l'esquema "seguir objectiu" es podria instanciar en "seguir company fins a la base sense xocar-hi" o "seguir presa fins a tocar-la", i aquestes dues instàncies serien dos processos diferents. Podem tenir, per tant, llibreries d'esquemes.
- Arbitratge entre esquemes: la coordinació s'aconsegueix mitjançant la fusió de comportaments individuals , és a dir, sumant i normalitzant les sortides individuals (vectors) dels esquemes [75]. D'aquesta manera els

comportaments s'agrupen de la manera més convenient proporcionant un sol vector de sortida (com s'observa a l'exemple de la Figura 14), i no tenim un model de capes (com en la *Subsumption Architecture*), sinó una xarxa dinàmica d'esquemes instanciats per un context concret [4]. Això dóna una major flexibilitat al model. Aquesta flexibilitat es deu a que no hi ha una única manera d'assolir els objectius. Per diferents condicions de l'entorn, la fusió de vectors de sortida dels diferents esquemes fa que predominin els comportaments associats a uns esquemes o altres. D'aquí que, seguint amb l'estratègia *bottom up* d'aquest model, puguin aparèixer comportaments d'alt nivell a partir d'un nombre limitat de comportaments de baix nivell (també anomenats comportaments emergents).

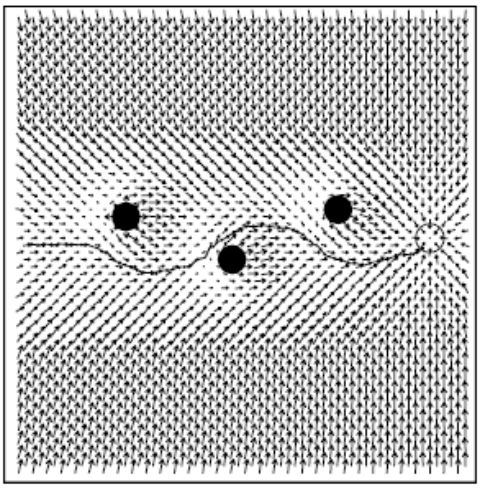


Figura 14. Camí generat amb la combinació dels vectors dels esquemes "*anar cap a l' objectiu*" , "*evitar obstacle estàtic*" i "*mantenir-se al camí*" [4].

- Comunicacions entre esquemes: la utilització d'un sistema de pissarra permet que els esquemes es comuniquin entre ells per evitar problemes com el *deadlock*.

Per altra banda, també el 1989, Lyons i Arbib [45] publiquen el seu treball on intenten desenvolupar un model computacional per a la programació de robots basant-se en les característiques específiques d'aquest tipus de sistemes. La seva intenció és proporcionar un model matemàtic, formal, més que un llenguatge de programació. Aquest treball formal serà la base per posteriors llenguatges de programació de robots. Per ells, les característiques específiques de la programació de robots són les següents:

Els programes de robots interactuen amb el món.

El paradigma central de la programació de robots és que l'entrada sensorial està enllaçada amb el coneixement per produir l'acció apropiada.

Els programes de robots mostren una estructura sensoriomotor flexible i jeràrquica.

Els programes de robots es defineixen de forma recursiva usant un *schema* o estructura de classe.

Els programes dels robots són inherentment distribuïts.

Les arquitectures basades en esquemes s'ajusten a aquestes 5 característiques, de fet Lyons i Arbib van anomenar *Robot Schema* (*RS*) al seu model formal. Aquest és un model distribuït amb un nombre d'agents interaccionant concurrentment. A *RS* un agent és una instància d'un esquema, utilitzant el terme "*Scheme Instance"* (SI) per denotar-lo. El procés d'instanciar crea una SI, li assigna valors inicials i la connecta amb altres SI. Per exemple "$Joint_{i=a"}$ és una SI de l'esquema "Joint" on hem donat valor "a" a la variable "i". Si només hi ha un paràmetre que es pot inicialitzar podem posar "$Joint_{a"}$.

Per tal de construir l'estructura sensoriomotor demanada pel punt 2, necessitem comunicar les SIs amb els altres. Cada SI té un conjunt d'objectes de comunicacions anomenats ports. Al fer la instància es poden especificar les connexions entre aquests ports i ports dins d'altres SIs. Per exemple, "T_V $(i_1,i_2,...)(o_1,o_2,...)$". Aquí *"V"* descriu com s'inicialitzen les variables del SI, les"i" són els noms dels ports d'entrada, i les "o" els noms dels ports de sortida. S'associa un tipus de dades a cada port. Només es poden enllaçar ports del mateix tipus. Hi ha comunicació quan un SI escriu un valor a un port de sortida que s'ha connectat, en el moment d'instanciar, a un port d'entrada d'un altre SI.

Cada esquema té associada una descripció comportamental que defineix com es comportarà una instància d'aquest en resposta a les comunicacions. En molts casos aquesta descripció serà una xarxa d'altres SIs. Usarem la notació $(A_i,...)^C$ per indicar que les SIs "Ai" estan connectades juntes tal i com es descriu al mapa de connexió port a port "C".

Més enllà d'aquestes especificacions d'esquemes bàsics i xarxes simples, *RS* permet fer assemblatges (*assemblages*) de xarxes i treballar amb elles com a mòduls bàsics, especificació de plans (amb suport al paral·lelisme d'esquemes, xarxes alternatives i sincronització amb l'entorn per mitjà de precondicions).

Per exemple, suposem que es vol especificar la xarxa de control de posició d'una articulació d'un robot, i el control d'aquesta ve donat per l'equació "$u_i=PC(x_i-x^D_i)$", on "u_i" és el senyal de control per l'articulació "i", "x_i" és la posició actual, "x^D_i" és la posició desitjada i PC és una rutina de control de posició. En *RS* aquesta tasca es representa amb una xarxa de tres SIs. "Jpos" és l'esquema que llegeix posicions de l'articulació, amb la variable interna "*joint*" que determina de quina articulació llegeix. Per tant "$Jpos_{joint}=i$" (o, simplement "$Jpos_i$") és una instància de "Jpos" per llegir de l'articulació núm. i. Aquest esquema només té un port de sortida, anomenat "x" pel qual escriu contínuament la posició actual, ho denotem "$Jpos_i()(x)$". També hi ha l'esquema "Jmot", també amb la variable interna "*joint*" referint-se a quina articulació està associat, el qual contínuament accepta un senyal de motor "u_i" pel seu únic port d'entrada i l'envia directament al motor, ho denotem "$Jmot_i(u)()$". Finalment, el tercer esquema s'encarrega de la connexió entre l'entrada sensorial i l'acció. Una SI de l'esquema "Jset" llegeix contínuament el valor del seu port d'entrada ("x") i escriu "$PC(x_i-x^D_i)$" al seu port de sortida ("u").

La posició desitjada ("x^D_i") es passa com a paràmetre en el moment d'instanciar l'esquema. Finalment, la xarxa d'esquemes en $\mathcal{RS}$ és la mostrada a la Figura 15:

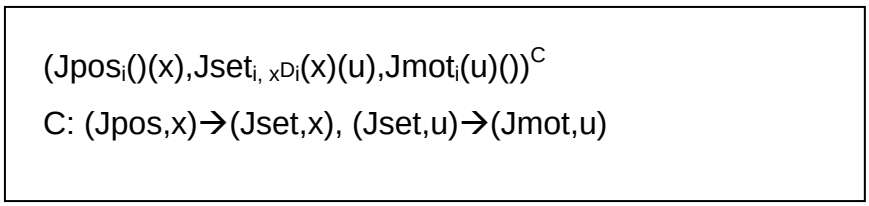


Figura 15. Xarxa d'esquemes en $\mathcal{RS}$

2.8.4 Arquitectures híbrides

Hem vist que les arquitectures SPA presentaven 2 grans problemes [31]: en primer lloc els canvis en l'entorn durant el procés de planificació, o bé durant el procés de modelat del món invalidaven els plans. I en segon lloc, els imprevistos durant l'execució d'un pla podien provocar que la resa de passos del pla s'executessin en un context erroni. Per altra banda les arquitectures reactives no presentaven aquests problemes, però la seva mancança d'estat intern, de representació del món presentava altres problemes, sobretot amb les tasques complexes que s'havien de planificar més enllà de la seva simple execució, però també amb altres tasques més simples.

Per exemple, la navegació reactiva utilitzant camps potencials pot presentar problemes de mínims locals. Un altre exemple ens el donen els errors de lectura dels sensors: en ocasions algun obstacle "desapareix" degut als errors de lectura. Si el robot tingués una mínima representació interna del món podria deliberar i arribar a la conclusió que aquella paret que ha vist abans quan ha passat no pot haver "desaparegut".

D'aquí que l'evolució natural anés cap al desenvolupament d'arquitectures que combinessin components tant reactius com deliberatius. Aquestes arquitectures híbrides es solen caracteritzar per estar estructurades en capes, amb les habilitats reactives als nivells inferiors i les capacitats deliberatives als nivells superiors. Durant els 90s diversos grups d'investigadors, més o menys independents, van arribar a solucions similars, amb arquitectures formades per tres components principals: les arquitectures de 3 capes (*Three-Layer Architectures*) [32], on un mòdul seqüenciador connecta la part deliberativa lenta amb la part reactiva. Utilitzant la terminologia de Gat per a l'arquitectura Atlantis (veure l'esquema de la Figura 16) [31][32], aquestes tres capes són:

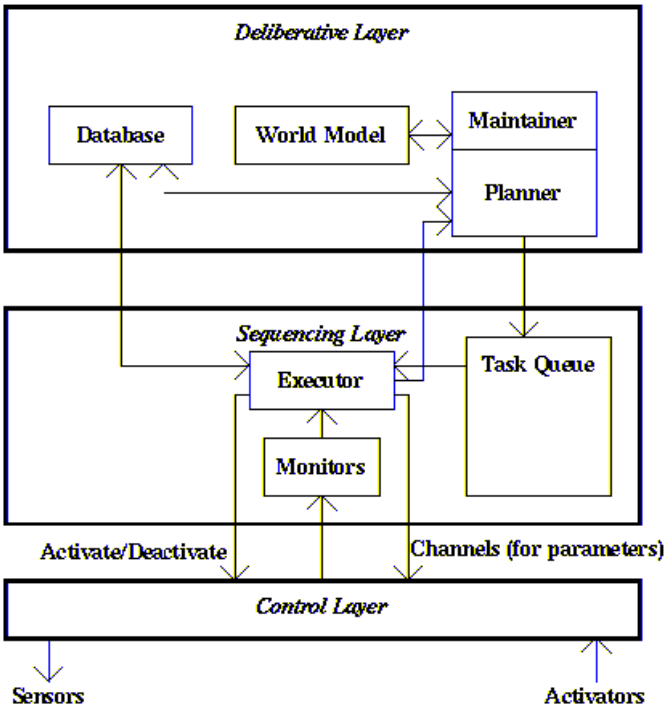


Figura 16. Atlantis [31], arquitectura de 3 capes

- El controlador: un o més *threads* enllacen sensors amb actuadors. La funció de transferència d'aquests es pot canviar en temps d'execució. Normalment conté una llibreria de funcions de transferència simples, anomenades primitives, que s'activen en funció d'alguna entrada del controlador. S'intenta evitar la utilització d'estat intern, o bé que aquest sigui mínim i efímer.
- El seqüenciador: La seva feina és triar els comportaments primitius que el controlador ha d'utilitzar a cada moment, i proporcionar paràmetres als comportaments. La dificultat aquí es deu al fet que la primitiva a executar en cada moment s'ha d'escollir en funció de la situació actual, per tant no és viable una seqüència lineal i simple. Una opció és enumerar tots els possibles estats en que es pot trobar el robot i calcular prèviament la primitiva a usar en cadascun d'aquests estats, però això només és viable per alguns dominis restringits si es codifica bé. Bàsicament presenta dos problemes. En primer lloc el robot no sempre sap en quin esta tes troba. Sobretot si passen coses imprevistes i, en segon lloc, aquesta estratègia desatén la "història" d'execució del robot, que sol contenir informació útil. Una alternativa és Una alternativa és utilitzar el *conditional sequencing*. Aquesta solució proporciona un model més complexa de plans d'execució, motivat per la capacitat humana de resoldre tasques complexes (inclús amb imprevistos) basant-se en instruccions molt precises. Aquí les construccions utilitzades per fer composicions de primitives no es limiten als condicionals, bucles i ordenacions parcials, sinó que s'han ampliat amb

construccions per suportar múltiples tasques interactuant en paral·lel. Hi ha llenguatges específics per implementar el *conditional sequencing*.

- El deliberador: aquí hi ha els algorismes que requereixen més temps de computació, normalment tasques de planificació, recerca exponencial, i algorismes de complexitat polinòmica amb constant molt gran (per exemple alguns de procés d'imatges). La característica arquitectural més important del deliberador és que mentre s'executen algorismes de deliberació (el deliberador s'executa en un o més *threads* independents) es segueixen executant transicions comportamentals (comportaments al nivell baix). Pot interactuar amb les altres capes del sistema de dues maneres diferents, produint plans perquè els executi el seqüenciador, o bé responent peticions específiques d'aquest (Atlantis segueix la segona opció), però no són opcions excloents entre elles.

Altres arquitectures híbrides presenten estructures conceptualment similars a l'anterior tot i no estar formades per tres capes. Una d'aquestes arquitectures, mostrada a la Figura 17, és AuRA[4] (*Autonomous Robot Architecture*), formada per dues capes o components principals: el sistema de planificació i el sistema reactiu. En aquest cas la component de planificació conté la part deliberativa en forma de sistema jeràrquic de 3 capes:

Mission planner: estableix els objectius d'alt nivell del robot i les restriccions en que ha d'operar. A les primeres implementacions aquesta component bàsicament ha actuat d'interfície amb un operador humà.

Spatial reasoner: utilitza coneixement cartogràfic guardat a la memòria per construir una seqüència de recorreguts en etapes que el robot ha d'executar per completar la seva missió.

Plan Sequencer: tradueix cada recorregut generat pel "*Spatial reasoner*" en un conjunt de comportaments motor per executar.

Passat aquest punt, la col·lecció de comportaments generats pel *Plan Sequencer* s'envien al robot per executar, aquí acaba la deliberació i comença l'execució dins del sistema reactiu. La part reactiva (*schema controller*) és la responsable de controlar i monitoritzar els processos comportamentals. Cada esquema motor s'associa a un esquema perceptiu capaç de proporcionar-li els estímuls requerits per aquest comportament particular. Un procés ("*move-robot*") s'encarrega de sumar i normalitzar els vectors generats pels esquemes i els transmet al sistema de baix nivell per l'execució.

Un cop comença l'execució reactiva, la component deliberativa no es reactiva si no hi ha una fallada a l'execució de la part reactiva. En aquest cas es torna a invocar el planificador jeràrquic.

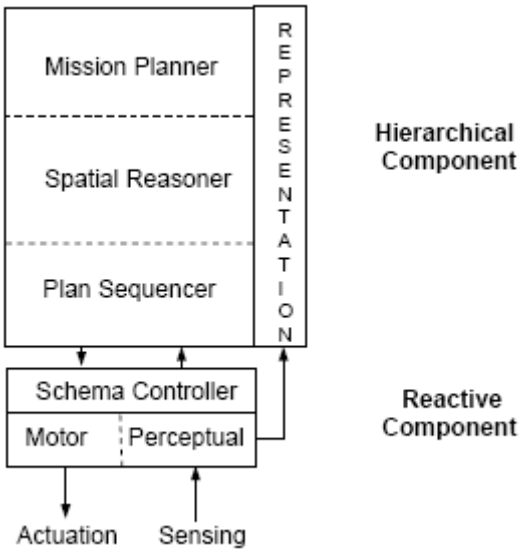


Figura 17. Arquitectura híbrida AuRA [4]

2.9 Utilització de la simulació

En un context com el del problema introductori vist al capítol 1.1, heterogeni i multidisciplinari, pot ser gairebé imprescindible l'ús de la simulació com a eina de validació, verificació, disseny i d'ajuda a la presa de decisions. Tot això en dependrà del punt o el nivell del sistema que es vulgui simular. En qualsevol cas, abans d'entrar en possibles aplicacions al nostre cas, s'ha de tenir present que la simulació sempre és una eina que ens ajuda, mai hauria de ser un objectiu en si mateixa. Cal tenir això en compte perquè, tot i que alguns aspectes d'una simulació es poden utilitzar fora d'un projecte concret, la construcció d'un model de simulació és un projecte en si mateix. Sobretot en el cas de la producció industrial. Per això, si el procés de modelat ha de ser prou costós i no es té clar l'objectiu d'aquest (quins aspectes ens interessen estudiar) potser hi ha altres solucions millors (solucions analítiques, implementació de prototipus, etc.). Finalment, un cop implementar el model, caldria verificar-lo (detectar possibles errors) i fer-ne la validació (verificar si produeix resultats realistes, i sota quines condicions) [22].

Es pot considerar que el procés de simulació no és més que "executar" un model de simulació, tenint en compte que aquest model de simulació és un conjunt de regles (màquina d'estat sistema d'equacions, autòmat cel·lular,...) que defineixen com anirà canviant el model a mesura que passi el temps [12]. És en funció de com s'obté la informació d'aquests canvis al llarg del temps que es fa la distinció entre models de simulació discrets o continus. En el primer cas aquesta informació s'obté en intervals de temps concrets (triats o donats pel propi sistema), i en el cas continu podem recuperar l'estat del sistema per qualsevol instant de temps (normalment mitjançant la utilització d'equacions diferencials). Es poden fer més subdivisions per classificar, per exemple, dins dels models discrets

també es podrien distingir segons siguin orientats a events, a procés o orientats a agent. En el cas dels events, aquests no són més que canvis que es produeixen en alguna variable del model (més enllà del propi temps) i això comporta canvis d'estat. Els models orientats a procés es composen d'un conjunt de processos o etapes, amb atributs ben definits, pels quals ha de passar un conjunt d'objectes (per exemple una seqüència de processos de producció). Finalment els models de simulació orientats a agent, on, a diferència dels anteriors, el modelat és *bottom-up* i es composa d'un conjunt d'agents com a models de les unitats bàsiques de simulació. Una darrera possible distinció es fa entre models deterministes, on es pot predir el proper estat en funció de les entrades, i models estocàstics, on això no és possible i apareixen els conceptes de distribució i aleatorietat.

A més de la distinció anterior entre tipus de models de simulació, el tipus de simulació que utilitzarem dependrà en bona part del nivell d'abstracció utilitzat per al modelat del sistema. Això es deu a que, per un mateix sistema, els diversos aspectes que volem estudiar apareixen a partir de certs nivells d'abstracció a partir dels quals ja no cal perdre recursos refinant el model. Per exemple, la simulació de protocols d'interacció entre agents no requereix modelar detalls de baix nivell de xarxa (latències, codificació binària, etc.). Però per analitzar programes de control numèric sí que és necessari un baix nivell d'abstracció per contemplar paràmetres de la màquina eina. A la Figura 18 es mostren diverses aproximacions a la simulació en funció del nivell d'abstracció i el concepte de continuïtat.

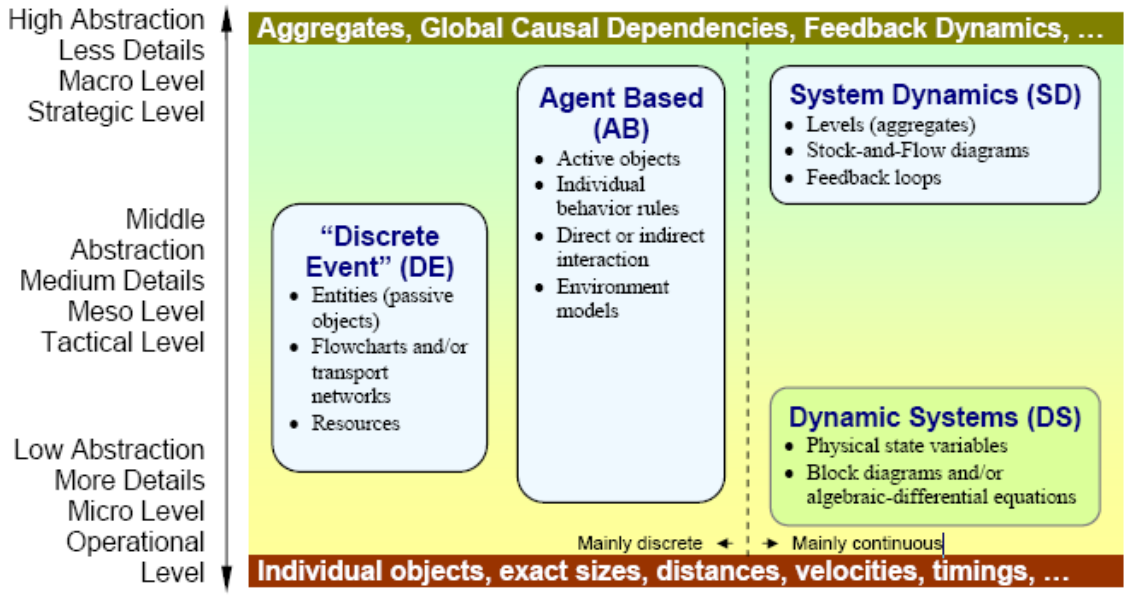


Figura 18. Paradigmes del modelat per simulació segons el nivell d'abstracció [12]

En general, els avantatges que ens pot proporcionar la simulació en un exemple com el nostre són:

- Ajuda al disseny: el modelat d'una planta de producció sencera abans de la seva construcció permet prendre decisions pel que fa a la distribució de les cadenes o cèl·lules de fabricació, *buffers* i magatzems, etc.
- Verificació d'estratègies de control i presa de decisions: sobretot en el control del procés productiu, com per exemple veure què passaria fent un canvi al sistema de gestió de magatzems, o canviant les assignacions de *buffers* locals, decidir si es poden acceptar més comandes,etc.
- Planificació de les tasques de producció: podent simular en pocs segons tota la feina d'un dia, es poden detectar els moments en que els recursos seran insuficients, les tasques aniran amb retard respecte la data prevista, etc. També es poden respondre preguntes del tipus "i si ...?", com per exemple saber què passarà si s'espatlla un recurs o n'augmenten la quantitat.
- Verificació i incorporació de nous productes de producció: per donar una resposta ràpida a les necessitats de fabricació de nous productes (un dels requisits de les noves empreses) es poden validar les dades de producció associades als nous elements, (plànols i especificacions de les peces, fulls d'operacions de producció, recursos requerits, llistes de materials, programes de control numèric, ...). mitjançant la simulació del seu procés productiu. Aquesta simulació asseguraria que es disposa de totes aquestes dades de producció necessàries per la fabricació de l'element i, a més, que aquesta informació està en el format adequat (això suggereix que disposarem d'una integració dels orígens d'aquesta informació, un altre dels requisits inicials). D'aquesta manera es poden modificar els documents, o sol·licitar-los al client abans de la fabricació real, assegurant un nivell de qualitat d'acord amb aquesta documentació. Un altre aspecte important de la verificació dels nous productes són els programes de control numèric (NC), la simulació permet detectar possibles errades i optimitzar els programes. La verificació tradicional dels programes NC, que consisteix bàsicament a repassar els recorreguts que farà l'eina sobre plànol i l'execució en màquines sense utillatge, no detecta errors que poden causar poca qualitat, envelliment prematur i trencament dels utillatges, etc. Simulant l'execució sobre un model de màquina es poden detectar errors a la geometria de peça generada, col·lisions, distància recorreguda per eines de tall, etc [70].
- Simulació de robots i MRS: els entorns industrials estan normalment bastant acotats (no són espais per explorar, en coneixem els elements principals, hi ha elements de seguretat, ...) i amb sistemes de transmissió de dades fiables. Però segueixen sent entorns molt dinàmics i heterogenis on els imprevistos que alteren els plans inicials es poden succeir (comandes de darrera hora, avaries als recursos, intromissions diverses,...). Per tant, la visió dels diversos elements de la planta com a sistema multirobot (MRS) ens permet analitzar i solucionar diverses

qüestions mitjançant els paradigmes, eines i metodologies disponibles pels MRS. Per exemple, les plataformes de simulació mitjançant agents faciliten la migració de l'entorn simulat cap a agents robòtics físics [66]. En funció del nivell d'abstracció podem tenir dos tipus de simuladors de robots: simuladors físics, i simuladors de robots abstractes [47]. La simulació de robots en entorns simulats (simuladors de robots a nivell físic) pot servir per prendre decisions de disseny dels robots físics. Si l'entorn simulat s'ajusta a l'entorn físic es poden prendre decisions de disseny basades en els resultats de la simulació. Exemples de decisions: tria de sensors, la seva col·locació, avaluació del sistema sencer, proves amb diversos models robots, tria d'utillatges de tall, etc. Per altra banda, la simulació de robots abstractes, serveix per experimentar amb aspectes no relacionats amb la interacció amb el medi. Normalment s'usa un alt nivell d'abstracció en les relacions amb l'entorn. Per exemple, es poden tenir primitives tipus "reconèixer objecte". Des del punt de vista de la IA distribuïda la simulació d'agents abstractes pot servir per testejar els models de cooperació i comunicacions entre els elements robòtics del grup. Per exemple protocols de comunicacions, estratègies per evitar conflictes i *deadlocks*. Aquest tipus de simulació és molt important en un sistema distribuït i heterogeni perquè la complexitat dels comportaments emergents que poden mostrar fa que, per preveure'n el comportament, no ens serveixin (o siguin insuficients) les eines analítiques tradicionals. En aquest cas, la simulació pot ser l'eina requerida per generar i validar el comportament de grup [20][73]. A més, algunes tècniques d'aprenentatge requereixen experiència i evolució de l'agent en l'extorn, és a dir, entrenament. La simulació permet fer aquest entrenament de forma virtual i accelerada. Per exemple, en algorismes evolutius o xarxes neurals es poden simular milers de generacions en poc temps [46].

- Integració de la simulació i suport a la modularitat: fins ara s'ha utilitzat la simulació com una eina *stand-alone* per a l'anàlisi i planificació, però l'eficàcia d'aquesta eina augmenta si es connecta amb el sistema de producció. S'han fet treballs on la fusió d'informació provinent de múltiples punts del sistema (per exemple en forma d'events) es connecta a un model de simulació del sistema. Això permet estudiar possibles escenaris a partir de la situació actual de la producció, monitoritzar sobre el model les operacions de la maquinària, desenvolupar codi de control temporal necessari en cas d'imprevistos (manteniment de màquines, etc.) i, en cas de fallades importants, recuperar tot el codi de control, informació de sensors i informació diversa de la base de dades temporal de simulació per poder fer un "replay" de les darreres activitats [23]. Més enllà de la integració del simulador per a l'anàlisi de dades i planificació, la introducció d'elements simulats o "virtuals" al sistema pot ser útil a l'hora de fer-ne la verificació global o durant la fase d'integració dels mòduls heterogenis [73]. Per exemple, abans d'adquirir un nou recurs es podria connectar un model

simulat d'aquest (convenientment programat perquè també consumeixi recursos "virtuals") a la xarxa de comunicacions del sistema de planta per veure com afecta al sistema global, és a dir, que no causi una saturació o col·lisions al sistema de missatgeria, que no caigui en estats no previstos, etc. Un altre exemple seria introduir un grup de AGV simulats (per exemple amb eines de simulació de MRS) per saber si la seva programació és correcta i no hi haurà interferències amb altres elements i, a més, realment suposaran una millora al rendiment global. La simulació podria detectar que, encara que un AGV millorés el rendiment del sistema, la freqüència i volum dels missatges de coordinació dels AGV a la xarxa suposés un coll d'ampolla. Finalment, en sistemes que es transformin progressivament cap a sistemes productius basats en MAS, la simulació esdevé una eina fonamental per validar els agents que s'implementaran, ja que permet, d'una banda, poder treballar amb el sistema complet a alt nivell i, de l'altra, emular la resta del sistema per als agents ja existents.

3. Plataformes multiagent

3.1 Introducció

La visió del centre de producció del problema exemple com a un conjunt de mòduls autònoms, amb capacitats i característiques diferents entre ells (heterogeni), molts d'ells mòbils i amb comportament proactiu fa que el model multiagent ens sigui útil en la nostra tasca. Això és degut, en primer lloc, a que aquest conjunt de característiques dels mòduls s'ajusten al model d'agent que hem vist al capítol 2.5.1) i, en segon lloc, aquests elements s'han d'integrar, comunicar i coordinar entre ells dintre d'un entorn dinàmic i imprevisible, cosa que ens porta cap al model multiagent. Aquí es mostren dues d'aquestes plataformes. Tot i que existeixen altres plataformes disponibles no s'ha continuat la recerca d'altres opcions per diversos motius [44]. No totes són *open source*, tampoc totes són *FIPA-compliant.* Un altre requeriment per un problema com el de la planta d'aquest treball és que ha d'existir una versió per dispositius mòbils, i que suporti múltiples protocols de xarxa. Finalment, algunes de les plataformes que podem trobar fa temps que no s'actualitzen i no hi ha activitats de manteniment als equips que les han implementat.

3.1.1 Zeus

Zeus [59] és un conjunt d'eines *open source*, implementades en Java, creades per British Telecom a finals dels anys 90. De fet, aquestes eines són el suport a algunes etapes de la metodologia proposada per Zeus per al disseny i implementació de sistemes multiagent col·laboratius. A més, també aporta la plataforma d'execució dels sistemes generats i diverses eines de visualització i *reporting*. Les components Java del *toolkit* de Zeus es poden dividir en tres grups funcionals (llibreries):

- *Agent component library*: és el conjunt de classes que formen les components bàsiques dels agents (*building blocks*). Aquestes components s'encarreguen de proporcionar la funcionalitat independent d'aplicació a nivell d'agent (planificació, comunicacions, ontologies, etc.).
- *Agent building software*: aquesta llibreria conté eines visuals que ajuden al desenvolupament dels agents específics per aplicació, ocultant així les complexitats de la *Agent component library*. *Agent Generator* n'és l'eina principal. Es tracta d'una *suite* d'editors integrats que permeten crear els agents mitjançant eines visuals, aquesta inclou un editor per editar l'ontologia del domini d'aplicació, un editor de variables per descriure instàncies específiques dels fets de les ontologies, l'editor d'agents que permet descriure els agents que formen part del sistema, un editor de

tasques per especificar-ne els atributs i fer composició de tasques, un editor de coordinació que permet triar els protocols de coordinació per cada agent, i un editor d'organització que permet especificar les relacions entre els agents. A la Figura 19 es mostra i l'editor d'ontologies integrat. Finalment s'integra un generador de codi que, partint de la informació entrada a quests editors, utilitza la *Agent Component Library per generar* el codi executable del sistema.

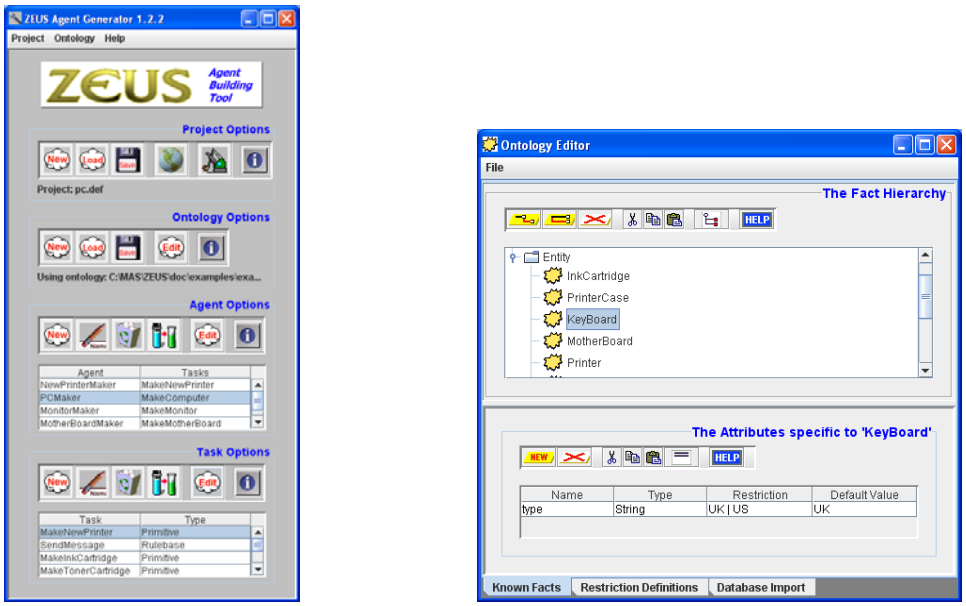


Figura 19. Eines visuals de Zeus: la suite Agent Generator (esquerra) i l'editor d'ontologies integrat (dreta)

- *Zeus utility Agents*: aquesta llibreria inclou tres components principals, que no són més que agents Zeus simples, implementades utilitzant la *Agent Component Library*. En primer lloc el servidor de noms (*nameserver*) al qual els agents d'una societat Zeus es registren al principi de la seva execució (li proporcionen la seva adreça). A més, el *nameserver* manté el rellotge de referència de la societat (si hi ha diversos *nameservers*, com que són agents normals el "temps zero" del sistema el tindrà el primer que s'ha executat). En segon lloc, el *facilitator* manté una base de dades amb les habilitats dels agents que pot ser consultada per aquests (i hi poden registrar les seves capacitats). Finalment, els agents visualitzadors permeten visualitzar o depurar les societats d'agents. Funcionen demanant als altres agents pel seu estat intern. Diverses eines de visualització aprofiten aquesta informació: el *society viewer* permet veure tots els agents i les seves relacions, una eina de *reports* per fer informes, una eina de control per controlar els agents remotament, etc.

Pel que fa a la metodologia [19] per Zeus, aquesta és similar a d'altres metodologies de disseny existents i es composa de les quatre etapes bàsiques que es mostren a la Figura 20:

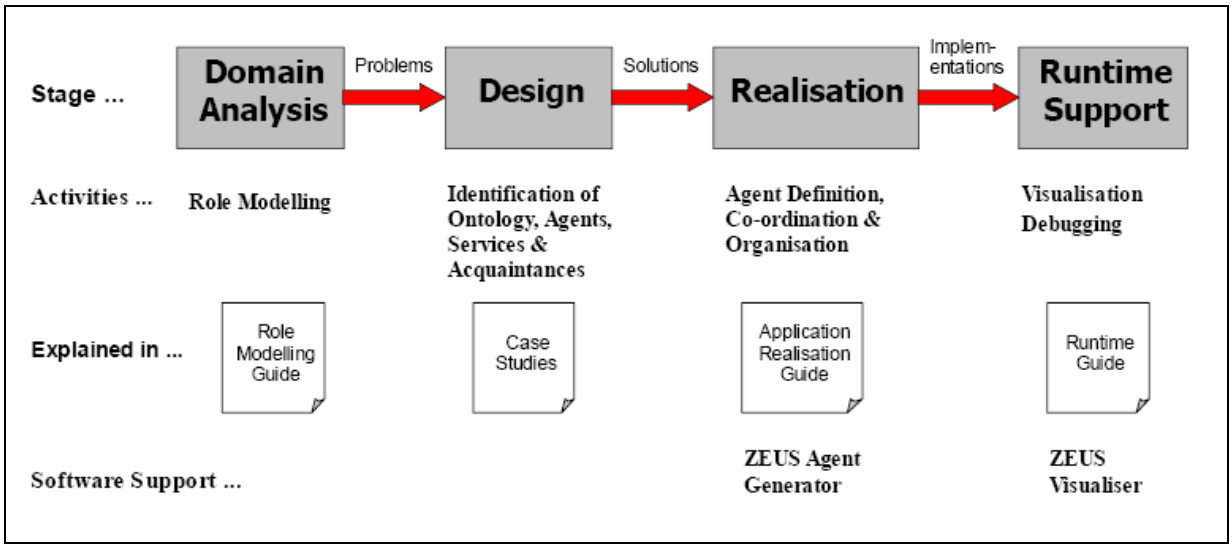


Figura 20. Metodologia de desenvolupament de Zeus [19]

- Anàlisi del domini: el propòsit d'aquesta primera etapa d'anàlisi és entendre el problema de l'aplicació. Aquí Zeus no proposa cap tècnica, ho deixa a llibertat del desenvolupador. De totes formes es proposa la utilització del *Role Modeling* per aquesta anàlisi. Com que aquesta tècnica és independent de la tecnologia d'implementació, aquesta a anàlisi es podria implementar amb qualsevol altra plataforma de desenvolupament. Zeus no té software de suport a aquesta etapa.
- Disseny dels agents: quan comença el procés de disseny el desenvolupador hauria de conèixer quins agents hi haurà al sistema i quines responsabilitats tindran. Aquí cal traslladar els rols trobats a l'etapa anterior als problemes que representen a nivell d'agent. Si a l'etapa anterior calia entendre els requeriments del sistema, aquí cal tenir traça i saber com reutilitzar i adaptar solucions existents. (Per això cada *Role Model* que aporten a la fase d'anàlisi porta un *Case Study* que descriu el raonament que hi ha darrera del disseny per una aplicació exemple). Zeus tampoc proporciona software per aquesta etapa.
- Realització dels agents: en aquest procés consisteix en la implementació dels agents a partir dels dissenys conceptuals creats a l'etapa anterior. La creació dels agents es composa de diverses etapes, estretament relacionades amb els nivells d'abstracció dels agents ZEUS. És a partir d'aquesta etapa en que Zeus proporciona ajut amb el seu software (l'*Agent Generator*, que hem vist anteriorment).
- *Runtime*: Zeus dóna suport en temps d'execució amb els agents *Visualizer*. Això reflexa el fet que el procés de desenvolupament no finalitza amb la implementació dels agents, sinó que aquests després han de ser depurats, optimitzats i testejats.

Malauradament no hi ha actualitzacions, ni manteniment de la plataforma des de maig del 2001. A més, no hi ha una versió per dispositius mòbils. De totes formes

és un exemple interessant de metodologia i eines de suport que van més enllà del moment en que s'implanta la plataforma i s'executen els agents. Les etapes prèvies a la implementació (la utilització de la plataforma només és una part de la metodologia) també poden ser utilitzades com a metodologia per al desenvolupament d'aquest tipus de sistemes, independentment de la plataforma utilitzada.

3.1.2 JADE

JADE [38] (*Java Agent DEvelopment Framework*) és la plataforma més utilitzada actualment per al desenvolupament de sistemes multiagent. El Telecom Italia Lab. (Tilab) en va llançar la primera versió *open source* l'any 2000 sota llicència LGPL. Es composa d'una plataforma que compleix les especificacions FIPA i llibreries en Java per a la programació dels agents. A més, es proporciona una *suite* d'eines en entorn gràfic per monitoritzar i administrar la plataforma.

La plataforma *FIPA-compliant* on s'executen (o "viuen") els agents es pot implantar de forma distribuïda. D'aquesta manera una plataforma JADE es pot veure com un conjunt d'instàncies de JADE que s'executen sobre màquines virtuals de Java (JVM) en *hosts* separats. Per tant, cada JVM és un contenidor d'agents de la plataforma. Per complir les especificacions de plataforma d'agents FIPA [27] un d'aquests contenidors és diferent dels altres, el *main container*, per això l'existència d'un altre *main container* implica l'existència d'una altra plataforma.). A l'exemple de la Figura 21 es mostra un escenari amb dues plataformes, una amb tres contenidors i l'altra només amb un. El *main container* sempre ha d'estar actiu a la plataforma i és on s'executen dos agents especials [9][15]:

- *Agent Management System* (AMG): aquest agent supervisa i controla l'accés a la plataforma (per exemple, se li poden fer peticions per eliminar o crear agents en contenidors remots). A més proporciona el servei de noms, assegurant que els noms dels agents són únics a la plataforma.
- *Directory Facilitator:* proporciona el servei de pàgines grogues, mitjançant el qual els agents poden trobar altres agents que proporcionin els serveis que necessiten els primers.

El *main container* sempre és el primer que s'executarà a la plataforma i els altres s'hi ha de registrar quan es posen en execució. Per això han de saber com trobar el seu contenidor principal (per exemple proporcionant-los el *host* i el port. Aquesta distribució dels contenidors en màquines diferents, però formant una plataforma per a tots els agents dels contenidors fa que els mecanismes subjacents de transport quedin transparents a la implementació del MAS. Així, per exemple, es pot seleccionar el protocol de transport en temps d'execució (Java-RMI, HTTP, IIOP, ...).

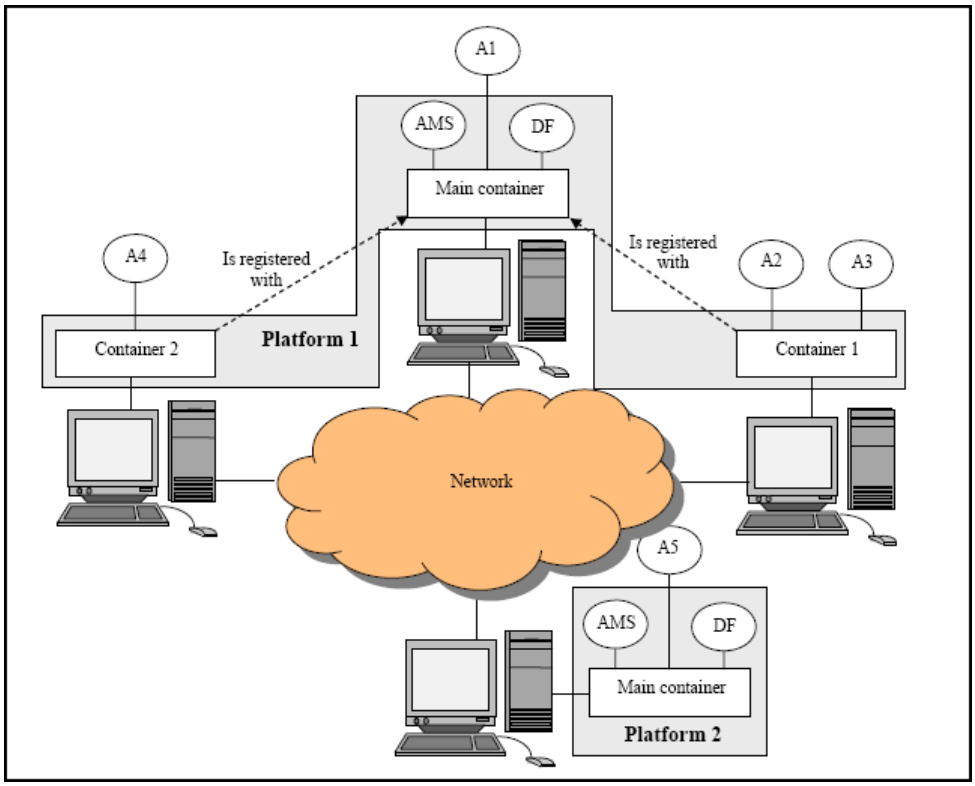


Figura 21. de JADE amb diversos contenidors que formen dues plataformes [15]

Pel que fa al model de comunicacions de FIPA, aquest està completament implementat als mòduls de JADE i les llibreries de programació faciliten el tractament dels següents aspectes:

- Implementació dels agents: per fer-ho només s'ha de derivar la classe *Agent* i s'accedirà a la funcionalitat dels agents JADE amb mètodes per enviar missatges, sol·licitar un identificador per a l'agent,codi d'inicialització, afegir comportaments etc.
- Comportaments: les tasques que pot realitzar un agent es programen utilitzant o derivant la classe *Behaviour*. Només cal crear alguns d'aquests objectes i afegir-los als comportaments de l'agent amb el mètode *Agent.Addbehaviour*. Les tasques es poden agrupar i ordenar formant comportaments complexes compostos. Per fer-ho es proporcionen, entre d'altres, classes per facilitar l'execució de tasques en forma seqüencial (*SequentialBehaviour*), en paral·lel (*ParallelBehaviour*) i, fins i tot, màquines d'estats (*FSMBehaviour*). Aquests comportaments complexes també ofereixen mecanismes de sincronització (esperar que acabin tots, acabar quan n'acabi un, etc.).
- ACL: la classe *ACLMessage* permet encapsular el contingut del missatge (en principi no és obligatori que aquest estigui en cap *Content Language* específic) i associar-li propietats per al seu tractament. Algunes d'elles són la ontologia utilitzada, tipus de codificació del contingut, l'*interaction-protocol* del que forma part el missatge i la performativa o intencionalitat del missatge (INFORM, AGREE, SUBSCRIBE, etc.).

- Protocols d'interacció: JADE proporciona classes per implementar alguns dels protocols d'interacció FIPA (*Request-Inform, Contract-Net, Subscribe, Propose*,..), però no està limitat a aquests. Només cal derivar una d'aquestes classes i omplir el codi de les funcions que gestionaran (*handlers*) les possibles situacions del protocol. Per exemple, la funció *handleRefuse(ACLMessage msg)* contindrà el codi per gestionar la situació en que el missatge "*msg*" és un "refuse".
- Suport per ontologies i llenguatges de contingut: JADE permet definir i usar llenguatges de contingut (CL) basats en ontologies definides externament. Aquí podríem distingir entre la part de llenguatge de contingut (JADE suporta FIPA-SL, LEAP i Java Codec) i la ontologia del contingut (que descriu l'estructura i la semàntica del contingut del llenguatge). Aquest suport al tractament del contingut del llenguatge facilita la conversió del conjunt de bytes del contingut cap a un conjunt de classes Java (i a l'inrevés). JADE s'encarrega de fer aquestes conversions amb el *ContentManager* (cada agent té un ".*getContentManager*()"). De fet el *ContentManager* delega les conversions a la ontologia i el Codec. Amb més detall, la Ontologia valida la informació des del punt de vista semàntic i el *Codec* fa les transformacions en cadenes de caràcters o seqüències de bytes seguint les regles sintàctiques del CL. Aquí és on trobem el suport per al tractament i codificació de continguts en XML, per exemple.

A més de tots el mòduls que incorpora el *framework*, al ser una plataforma *open source* es disposa de diversos *plug-ins,* utilitats *open* source i versions de JADE adaptades a entorns específics. Aquí podem destacar l'editor d'ontologies Protégé [67], que disposa del *plug-in BeanGenerator* [64] per transformar les ontologies en classes JADE. Si no es disposa d'un mòdul com aquest, el procés de desenvolupament de les classes que representen l'esquema, predicats, fets i conceptes pot ser molt costós. Pel que fa a versions específiques de JADE, cal anomenar JADE-LEAP [16]. Aquesta versió permet executar JADE (i, per tant, agents FIPA) sobre dispositius mòbils com telèfons, PDA, etc. A més, la versió *dotnet* permet executar JADE sobre màquines on s'executa .NET Framework [52]. Anant encara més enllà, per "sobre" de JADE, al Tilab han desenvolupat una altra plataforma *open source*, WADE [80], basada en JADE per a l'execució de *workflows* (fluxos de treball). En aquest cas han derivat l'agent bàsic de JADE per implementar un motor d'execució de fluxos de treball. Per tant, a diferència d'altres entorns amb un potent motor d'execució de fluxos, aquí podem tenir diversos motors més lleugers (agents) que s'encarreguen d'executar en paral·lel parts concretes dels fluxos de treball més complexes. Això pot ser interessant, per exemple en entorns d'empresa, per a l'execució de fluxos de treball basats en el model de processos del negoci. Totes aquestes característiques fan que JADE sigui la plataforma que més s'ajusta als requeriments del model de solució proposat en aquest treball.

4. El model teòric de planta

4.1 Introducció

Tal i com s'ha mostrat al capítol 1.5 abans d'iniciar el procés de transformació o adaptació dels subsistemes de la planta és necessari obtenir el model teòric de la planta. Cal disposar d'aquest model perquè permetrà planificar la planta i possibilitarà el disseny (comportament, validació i implementació) dels agents que es desenvoluparan. En el cas ideal aquest consisteix en un model complet i exhaustiu de l'empresa completa totalment agentificada. En aquest treball, però, s'exposa el model de la planta de producció en un primer nivell de refinament, mostrant només les entitats més importants del procés productiu i els processos relacionats amb la producció del producte final. Aquest model s'ha hagut de fer després d'una anàlisi de la planta perquè anteriorment no se n'havia fet cap model (o no en quedava constància). D'aquesta manera, mostrant els principals processos i tasques de producció, juntament amb les entitats encarregades d'executar-los, es pot fer una primera detecció de punts crítics, així com prendre decisions sobre propers canvis al model de negoci de la planta. Per exemple, el pas freqüent de missatges entre processos associats a entitats físicament separades pot centrar el treball en la necessitat d'incorporar entitats intermediàries encarregades d'alleugerir aquestes comunicacions, o bé implementar un tipus de comunicació no basada en el pas de missatges. També es poden detectar processos pels quals passen els fluxos de treball més importants del model, transformant-los en colls d'ampolla. A més, si s'opta per la utilització del model multiagent com a base de la solució, aquest model ajuda a prendre la decisió sobre quines són les entitats potencialment agentificables i els rols que els poden ser assignats.

A continuació es mostra aquest model de processos en notació BPMN [63], amb una explicació per cadascun d'ells. Per cada esquema, les entitats participants que es mostren sense processos són les que ja s'han vist en detall o es veuran als capítols següents.

4.2 Generació d'ordres de fabricació

Tot i que el sistema ERP inclou tota la gestió de l'empresa (compres, vendes, tresoreria, facturació, gestió d'estocs, etc.) aquest model es centra en la part associada al cicle de producció i, en concret, en la creació del model del producte, les ordres de fabricació i la seva gestió. Així i tot, en aquesta primera modelització no s'inclouen altres rols secundaris del sistema associats a la gestió de la producció, bàsicament consistents en la obtenció d'informació com, per exemple,

control del treball dels operaris, informes de l'estat del procés productiu o canvis en els temps concedits als processos. El resum d'aquest procés productiu es mostra a la Figura 22.

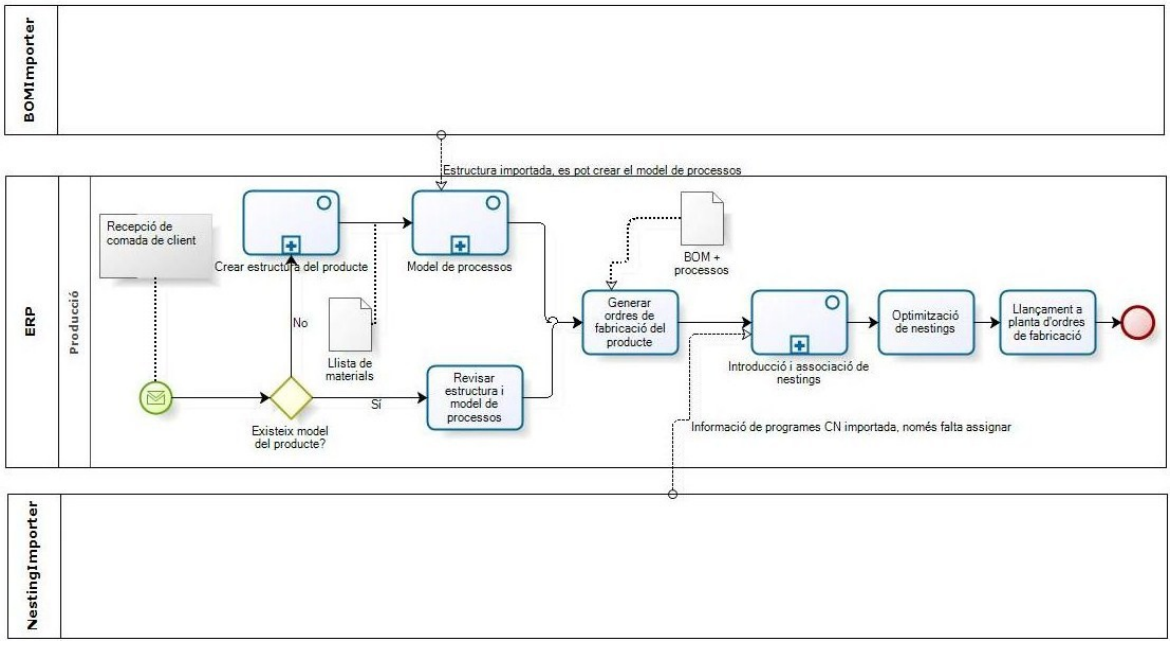


Figura 22. Esquema del cicle de fabricació dins del ERP

El cicle de producció del producte final s'inicia amb la recepció de la comanda del client. Tot i que no es mostra en aquest esquema, això afecta també a altres mòduls del sistema ERP (que aquí es podrien mostrar com a "rols" diferenciats). Per exemple, en funció de la comanda els mòduls d'estocs i compres s'encarreguen de verificar que l'estoc sigui suficient (segons la política de control d'estocs adoptada) i generar, si cal, les sol·licituds de compra de matèria primera corresponent.

Per generar les ordres de fabricació d'un producte final es necessita la informació que ens proporcionen el model del producte i el seu model de processos [18]. La creació d'aquests models es mostra a l'esquema en forma de dos processos seqüencials anomenats "crear estructura del producte" i "model de processos". A partir d'aquests es generaran les ordres de fabricació per a les parts de l'estructura del producte que siguin necessàries, cadascuna d'elles amb un pla de fabricació específic. El tercer procés important és l'encarregat, quan calgui, de buscar, associar o crear els programes de control numèric que executaran les màquines eina associades a les tasques del model de processos. Un cop fet això les ordres de fabricació poden ser "llançades a planta", és a dir, assignades als recursos corresponents sobre un calendari de fabricació. Els esquemes d'aquests tres processos bàsics per a la generació d'ordres de fabricació es mostren a continuació, expandint-los de la Figura 22.

4.2.1 Creació de l'estructura del producte

Aquest procés s'encarrega d'introduir el model del producte al sistema ERP tal i com s'observa a la Figura 23. Es tracta d'un esquema molt simplificat que permet mostrar la importància de la integració de mòduls i la utilitat del model de processos per detectar parts crítiques.

Figura 23. Creació del model estructural del producte

En primer lloc, el modelat del producte final es fa mitjançant eines CAD externes al ERP. Aquestes eines, més enllà del simple modelat 3D entrant valors dimensionals, permeten el modelat basat en característiques, anàlisi del comportament dels materials, llistes de materials i generació de fitxers CAM. Per tant, el primer procés de l'esquema consisteix en buscar aquest model CAD i generar informes de característiques. Posteriorment cal entrar la informació de cada element del producte (parlem de "peces" però es pot tractar de matèria primera o altres conjunts d'elements compostos). L'entrada d'aquesta informació per cada peça es representa en forma de procés iteratiu, indicant que no només s'ha d'afegir la informació a l'estructura d'informació, sinó que també s'ha d'afegir a la base de dades d'articles i elements del ERP en cas de no existir. Finalment, aquests elements (simples o compostos) que formen el producte final s'organitzen en una estructura jeràrquica, que també es troba al mòdul CAD. El resum d'aquesta estructura jeràrquica d'elements que composen el producte final, juntament amb les seves característiques físiques (pes, dimensions, tipus de material, etc.) es coneix com a llista de materials (*Bill Of Materials*, o BOM) [18]. A la Figura 22, a més, apareix l'entitat "BOMImporter", separada del ERP però que es comunica amb el procés encarregat del model de processos del producte Com que el procés de creació de l'estructura del producte dins del ERP és un procés bàsicament dut a terme pels operaris de la oficina tècnica, i la informació ja està al sistema CAD, aquest procés es pot assignar a una entitat diferent per tal d'automatitzar-lo. De fet és un rol clarament candidat a ser assignat a un agent del sistema.

4.2.2 Model de processos del producte

L'esquema de la Figura 24 representa els processos bàsics per a la creació del model de processos del producte. Per poder dur-los a terme cal disposar al sistema del model estructural que s'ha mostrat anteriorment.

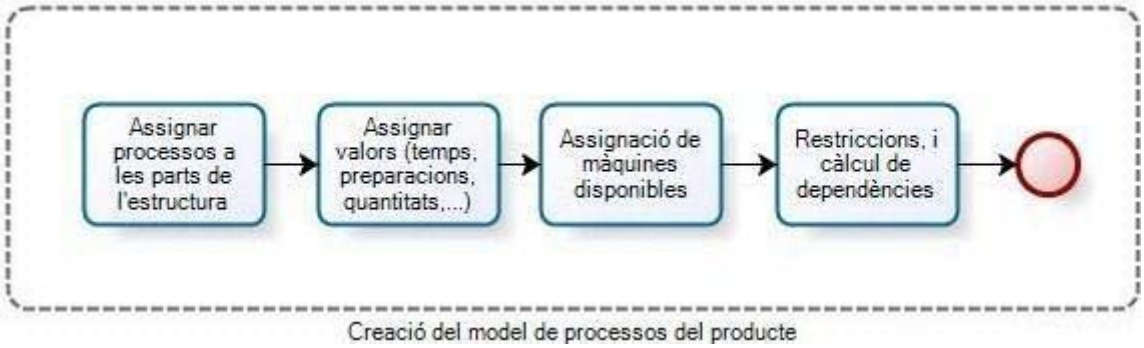


Figura 24. Associant processos de producció al model del producte

L'objectiu bàsic de la creació del model de processos del producte és associar als elements de la llista de materials el conjunt d'operacions (o "processos") de producció necessaris per a la seva fabricació. Posteriorment cal parametritzar aquests processos de producció amb els valors específics per produir la peça concreta. Aquests paràmetres depenen del procés concret, normalment es disposa de paràmetres tipus "temps de preparació", "tipus de soldadura", "temps unitari", "quantitat de recursos necessaris", etc. Naturalment cal associar als processos de fabricació el conjunt de màquines disponibles per realitzar l'operació. En aquest cas potser hi ha restriccions del client o dels enginyers sobre les màquines concretes que poden fer el treball (per exemple limitades a restriccions d'ample de tall, errors d'utillatge, etc.). A més, aquí també cal parametritzar els valors d'operació de les màquines per al procés concret de la peça (temps unitari, concedit, etc.). Finalment, un cop entrats els processos per cada peça s'associen restriccions de dependències entre aquests. Això es deu a que, a més de l'ordre natural entre els processos (tall, soldadura, pintura, etc.), alguns es poden sobreposar entre ells, no importar l'ordre d'execució o bé tenen restriccions en quant al temps o quantitat d'unitats produïdes en un procés abans de poder iniciar el següent.

Per tant, un cop s'ha entrat el model de processos ja es poden crear les ordres de fabricació del producte. Per cada comanda amb quantitat "Q" producte final que cal produir, es generaran "n" ordres de fabricació. Cada ordre de fabricació està associada a la producció (o processat) d'un element de la llista de materials del producte final, i la quantitat d'aquest element que cal produir en aquesta ordre de fabricació dependrà de "Q" i de la seva quantitat a l'estructura de la llista de materials.

Bàsicament, el model de processos (PM) per un producte final "P" es composa dels següents elements:

$$PM(P) = \{pm(E_i)\} : E_i \in BOM(P)$$

$$pm(E_i) = \{p_j\}$$

$$p_i = (M, d, DEP, ARGS)$$

És a dir, conjunt de processos *pm* per cada element *E* de la llista de materials del producte. Aquests processos p_i es composen del conjunt de màquines disponibles

M, una data objectiu (data límit de finalització del procés), el conjunt d'arguments de parametrització *ARGS* i el conjunt de dependències DEP. Per exemple, per a l'element *E*, podem tenir el procés p_5 que depèn de p_2 i p_3, però es pot fer al mateix temps que p_4. A més, es pot iniciar quan p_3 està al 75% (però ha d'esperar a que s'hagi acabat p_2).

Així doncs, el model de processos del producte no és més que un pla de fabricació per a cadascuna de les seves components.

4.2.3 Introducció i associació de nestings

Si bé el model de processos proporciona el pla de fabricació per a cada element de la llista de materials, algunes de les màquines que executaran aquests processos treballen amb programes de control numèric. A més, en el cas dels processos que treballen amb matèria primera es fa necessari reduir la despesa de material. Per això, abans de la generació del codi CN es fa necessari un procés de *nesting*. En el cas de les planxes de metall el nesting consisteix en maximitzar la superfície de planxa utilitzada a cada programa CN de les màquines de tall i punxonar. Com que no sempre és possible cobrir tota la superfície de la planxa amb repeticions de la mateixa peça, degut a les seves característiques geomètriques, s'intenta aprofitar la superfície amb la inclusió de peces que cal tallar o punxonar per altres ordres de fabricació (preferiblement del mateix producte final). Això s'aconsegueix utilitzant mòduls software específics que s'encarreguen de fer els càlculs corresponents a partir de les geometries importades de les eines CAD. Aquests mòduls generaran els programes CN a partir d'aquesta agrupació de peces (nesting). A l'exemple de la Figura 25 es mostra com s'aprofita al màxim la superfície produint 5 peces del tipus "5", una del tipus "10", una del tipus "6", etc.

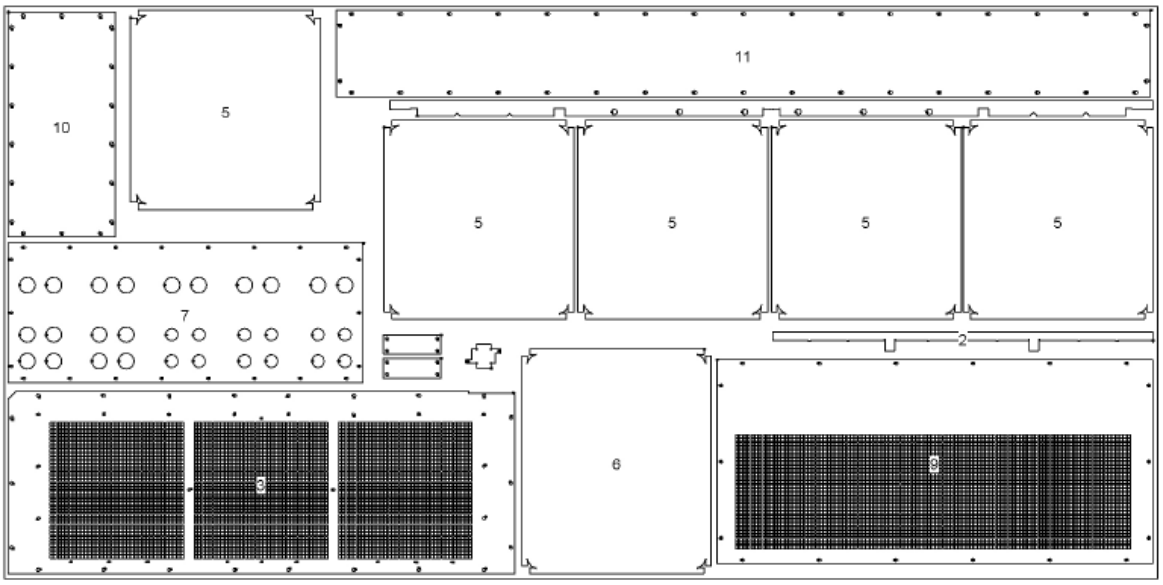


Figura 25. Exemple de nesting sobre una planxa de metall

Com que el procés de nesting pot dependre de les peces pendents de fabricar en un moment donat, és un procés que es fa un cop les ordres de fabricació estan generades però encara no han estat llançades a planta. Els mateixos mòduls de nesting permeten obtenir característiques més acurades de les peces un cop tallades o processades amb una màquina concreta, informació que també cal introduir al ERP. L'esquema d'aquesta seqüència de processos es mostra a la Figura 26. Igual que en el cas de la introducció del model estructural, aquí també s'ha previst la incorporació d'una entitat separada encarregada de passar la informació dels mòduls de nesting al sistema ERP (veure Figura 22).

Finalment, abans de llançar les ordres de fabricació a planta, i tenint ja els programes CN generats, a la Figura 22 es mostra un darrer pas d'optimització dels nestings. En aquest cas ja no cal optimitzar superfície a nivell de cada xapa, però sí que cal minimitzar la despesa de matèria primera en funció del nombre de vegades que s'executen els programes CN. L'exemple següent ajudarà a entendre en què consisteix aquest tipus d'optimització.

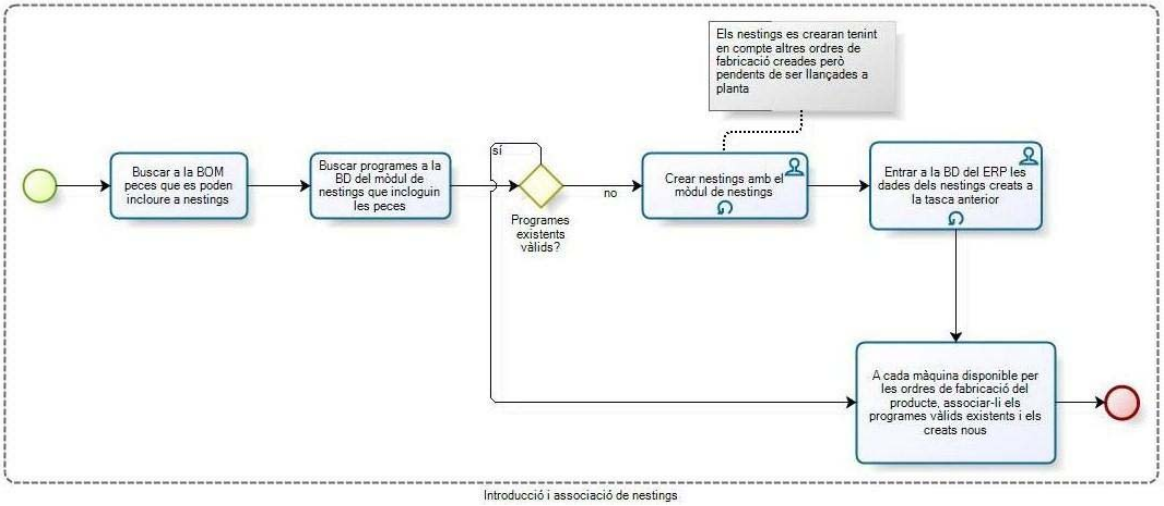


Figura 26. Creació i associació de nestings a les ordres de fabricació

Suposem que tenim ordres de fabricació pels elements E1, E2, E3, aquest elements s'obtenen del tall de planxa metàl·lica. Se n'han de produir 38, 75 i 47 unitats respectivament. El nesting d'aquestes 3 peces he generat 2 programes CN que optimitzen la despesa de la planxa. El programa P1 (de manera similar a l'exemple de la Figura 25) permet produir 3, 8 i 4 unitats dels elements E1, E2 i E3, mentre que el programa P3 en produeix 4, 6 i 5 unitats. Per tant, utilitzant 13 planxes amb el programa P1 a la màquina es produirien 39 elements E1, satisfent així les necessitats de producció. El problema és que, degut al nesting, també s'haurien produït 104 elements E2 i 52 elements E3. Evidentment, si no s'han d'utilitzar en posteriors muntatges, els 29 elements E2 produïts en excés són una despesa innecessària de matèria primera. Utilitzant el programa P2, també amb 13 planxes, es generarien 78 elements E2, quantitat més propera a la demanada, però hi hauria un excés dels altres dos elements (52 del E1 i 65 del E3). Per tant, caldrà intentar minimitzar l'excés d'elements produïts mitjançant la combinació de

programes, és a dir, utilitzar x_1 planxes amb el programa P1 i x_2 amb el P2. Això es pot resumir en el següent sistema d'inequacions:

$$3x_1 + 4x_2 \geq 38$$
$$8x_1 + 6x_2 \geq 75$$
$$4x_1 + 5x_2 \geq 47$$

En aquest cas, x_1=5 i x_2=6 donaria 39, 76 i 50 elements de cada tipus, però també seria vàlid x_1=6 i x_1=5. En aquesta segona solució els elements produïts són 38, 78 i 49. Per tant, la millor de les dues solucions serà la que minimitzi el pes de xapa utilitzada. Per tant, les inequacions anteriors no són més que les restriccions per a la minimització de la funció "Pes":

$$Pes = (3Pes_1 + 8Pes_2 + 4Pes_3)x_1 + (4Pes_1 + 6Pes_2 + 5Pes_3)x_2$$

On Pes1, Pes2 i Pes 2 són els pesos dels elements 1, 2 i 3.

Normalment, però, la situació serà més complexa, amb més tipus de peces per programa i una major quantitat de programes disponibles.

Per això aquest darrer pas d'optimització de nestings consisteix en trobar una solució òptima al problema de minimització d'una funció lineal "Pes" sota un conjunt d'inequacions que en són les restriccions. El resultat d'aquesta optimització s'introduirà al ERP perquè els operaris de màquina (o l'entitat del model encarregada de la operació de màquina) executin els programes corresponents el nombre de vegades necessari. En el cas de la planta exemple d'aquest treball el ERP incorpora un mòdul d'optimització lineal entera mitjançant el mètode del símplex enter [8]. L'exemple anterior es pot generalitzar:

Tenim *n* elements, i *m* programes.

$p_0 \ldots p_{n-1}$ són els pesos dels elements.

$x_0 \ldots x_{m-1}$ són les vegades que s'utilitzaran els programes a la solució.

n_{ij} és la quantitat d'elements *i* que es fan amb el programa *j*.

Es volen fabricar quantitats $q_0 \ldots q_{n-1}$ de cada element.

Volem minimitzar la funció:

$$Pes = (n_{00}p_0 + \ldots + n_{(n-1)_0}p_{n-1})x_0 + \ldots + (n_{0(m-1)}p_0 + \ldots + n_{(n-1)(m-1)}p_{n-1})x_{m-1}$$

Sota les restriccions:

$$n_{00}x_0 + \ldots + n_{0(m-1)}x_{m-1} \geq q_0$$
$$\ldots$$
$$n_{(n-1)0}x_0 + \ldots + n_{(n-1)(m-1)}x_{m-1} \geq q_{n-1}$$
$$x_0 \ldots x_{m-1} \geq 0$$

4.2.4 Importadors externs

Anteriorment s'ha fet referència a dues entitats que apareixen a la Figura 22 però no es mostra el seu detall de processos a l'esquema. Es tracta del "BOMImporter" i el "NestingImporter". Els processos de negoci associats a aquetes dues entitats es mostren a la Figura 27. No són entitats imprescindibles al model de planta, però poden dur a terme tasques que alliberen la càrrega dels dos punts crítics que formen la introducció del model estructural i la introducció de dades de programes i nestings (processos en part assignats a operaris humans i repetitius). Com que són esquemes prou simples, només cal observar que inicien la seva tasca amb la recepció d'un missatge i fan la importació de dades a partir de la base de dades de planta. Això vol dir que alguna altra entitat s'encarregarà d'avisar-los de l'existència de dades a convertir, i aquestes dades estan ja estandarditzades, independents del tipus de mòdul extern que contingui les dades de llistes de materials o nestings. Per tant, aquestes dues entitats poden seguir treballant independentment del tipus de mòdul CAD o CAM utilitzats.

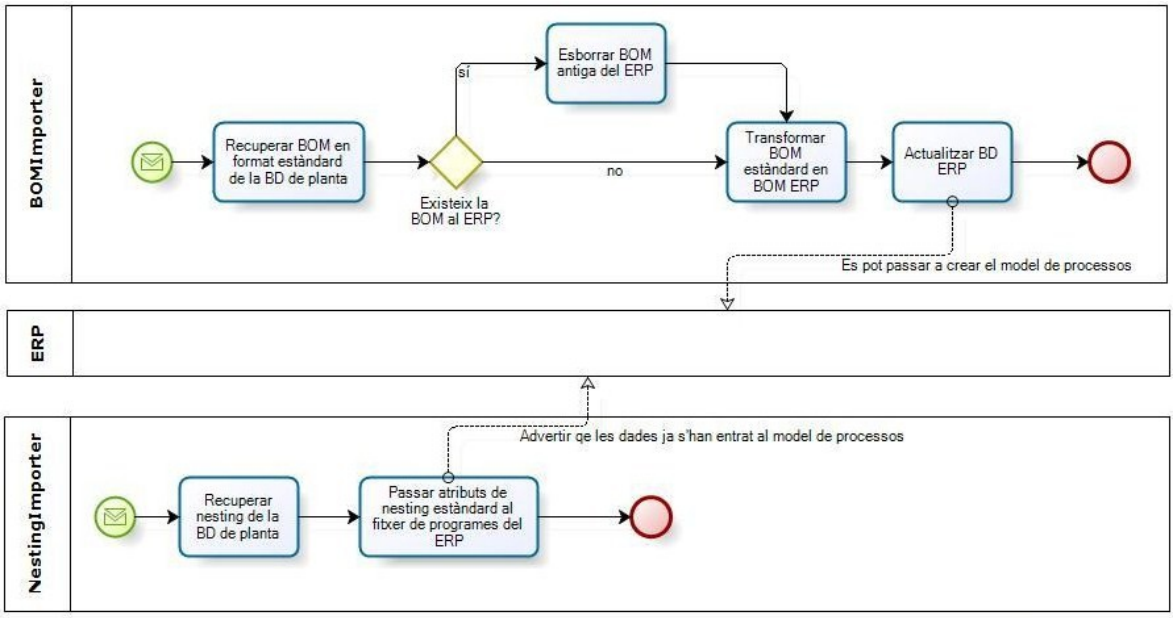


Figura 27. Importadors externs al mòdul ERP

4.3 Integració de sistemes software heterogenis

Fins ara, sobre el model de processos teòric de la planta, hem vist com la incorporació d'entitats externes permet establir un flux automàtic de la informació dels mòduls externs cap al ERP. També s'ha mostrat com aquestes entitats introdueixen informació al ERP independentment del tipus de sistema que ha generat aquesta informació. A més, es tracta de la integració de mòduls software i la informació s'intercanvia en forma de document. En concret, al model actual de planta, ens limitem a la integració dels mòduls CAD/CAM. En qualsevol cas, per a una futura integració d'altres sistemes llegats, es pot utilitzar el mateix mecanisme. A continuació es mostra la part del model de planta que permet al

"BOMImporter" i al "NestingImporter" incorporar dades independentment del seu origen.

Com que es vol fer la integració d'informació basada en documents heterogenis, cal incorporar al model de planta el concepte d'entitat traductora. Aquest tipus d'entitat s'encarrega de fer la traducció de la informació, fent de "pont" entre els dos sistemes. Això implica la existència d'una altra entitat encarregada de detectar les actualitzacions que fan els sistemes externs sobre els seus documents. Si s'utilitza la tecnologia multiagent per implementar aquest model, aquestes entitats es poden veure com a agents "LegacyWatcher" i "Translator", encarregant-se els primers "d'observar" els canvis al sistema llegat, i els segons encarregant-se de la traducció de la informació. Tal i com s'observa a la Figura 28, el LegacyWatcher rep, en forma de senyal, la notificació de canvis als documents externs.

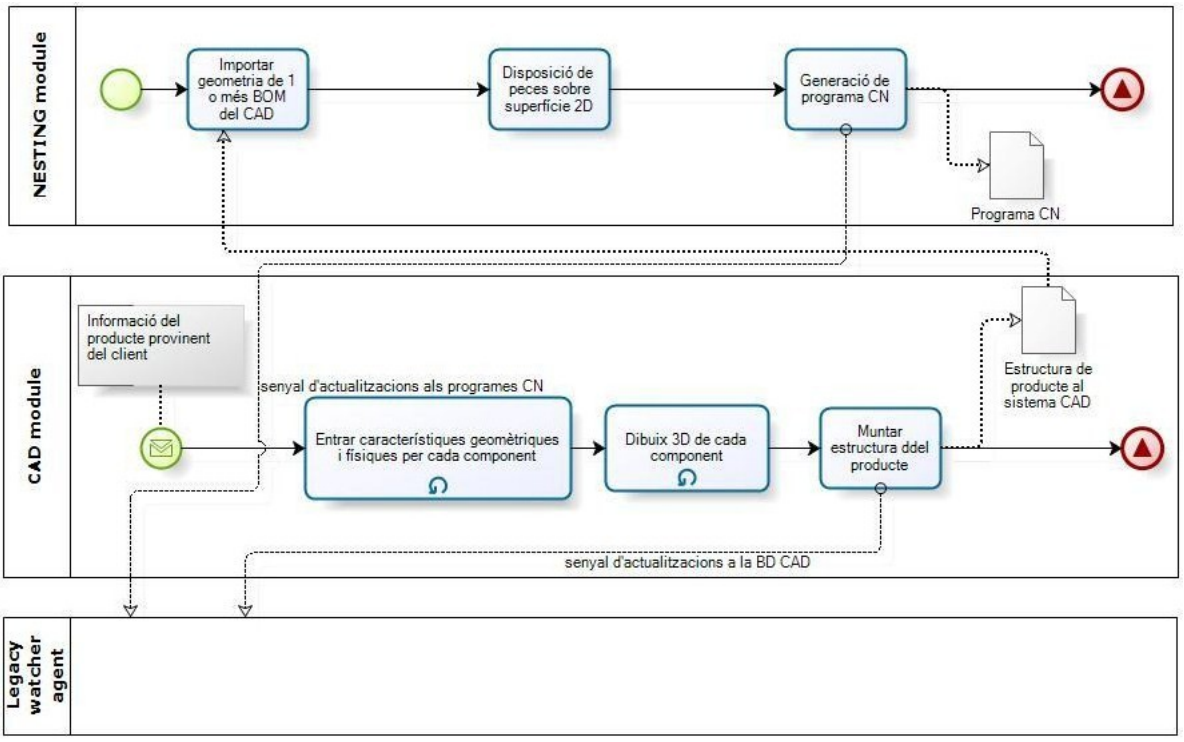


Figura 28. Detecció de canvis als mòduls CAD/CAM

El fet de representar el "LegacyWatcher" com a una entitat separada dels mòduls externs permet implementar-lo en forma de múltiples instàncies, una a cada màquina on s'executa el sistema legat, o bé una sola instància remota. També permet reutilitzar-lo per a la integració d'altres subsistemes ja que no fa cap conversió ni actualització, es limita a iniciar el procés d'integració. Així doncs, el model també incorpora una entitat a qui s'associaran els processos de traducció.

El pas següent a la detecció és la traducció de la informació, tasca associada a una entitat separada al model, anomenada "Translator". De la mateixa forma que en el cas del "LegacyWatcher", la representació en forma d'entitat separada permet tenir-ne diverses instàncies amb execució distribuïda. Es pot implementar com a mòdul que s'executa en una màquina dedicada, amb funcions de

traducció per cada tipus de document en funció de l'origen de les dades. Però també es pot implementar com a un conjunt d'agents, possiblement distribuïts, cadascun dels quals especialitzat en la traducció d'un tipus de document provinent d'un mòdul extern concret. Per això el model, que es mostra a la Figura 29, incorpora un "diàleg" o negociació entre el "LegacyWatcher" i el "Translator". D'aquesta manera el "LegacyWatcher" (poden ser diversos) pot buscar el traductor adient el tipus d'informació que cal convertir. El model permet que el "watcher" tingui una referència al traductor conegut si vol estalviar repetir sempre la negociació. En el cas d'utilitzar una plataforma multiagent estàndard com a *middleware*, es pot utilitzar algun protocol d'interacció estàndard com per exemple el *contract net*.

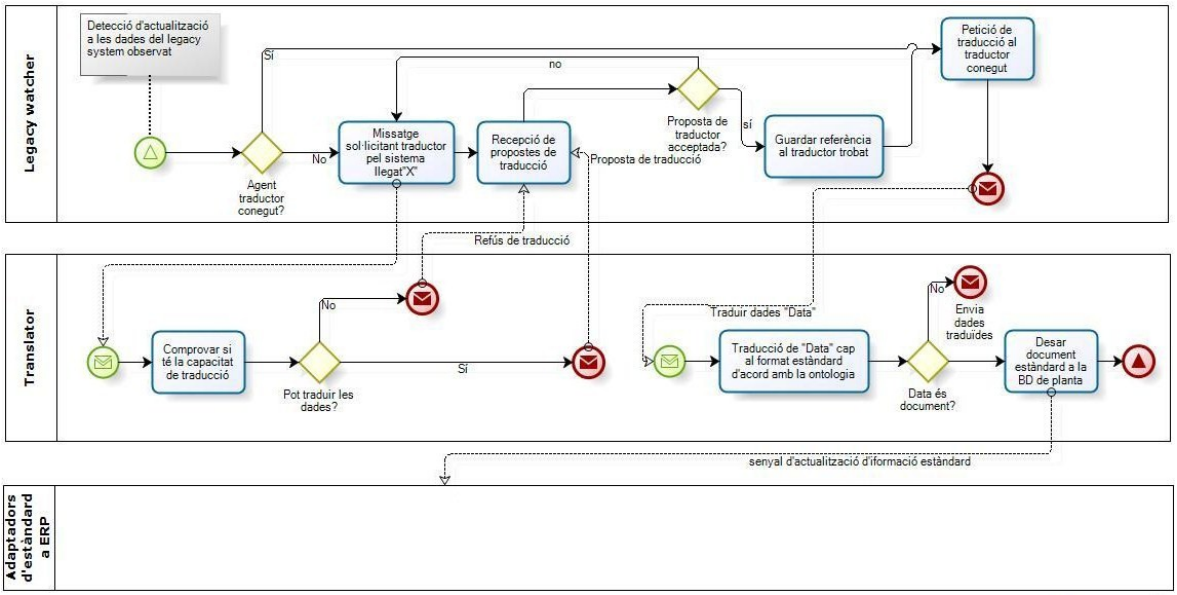


Figura 29. Processos de cerca de traductor i conversió de dades a estàndard

En general, el procés de traducció es pot resumir en dues etapes. Primer el "watcher" envia un missatge als traductors especificant el tipus d'informació a convertir, i aquests accepten o refusen la conversió. Entre les respostes rebudes, accepta el traductor adient i li envia l'ordre de traducció. En una segona etapa el traductor s'encarrega de la traducció.

Aquí torna a aparèixer el concepte de "base de dades de planta". El model permet que el "Translator" enviï dades traduïdes a una altra entitat directament, o bé que les desi a la base de dades de planta. La incorporació d'una base de dades global a la planta, que no existeix al model antic d'aquesta permet que els "translators" hi desin els documents que han convertit en un format estandarditzat, utilitzable per qualsevol subsistema que els pugui incorporar mitjançant els adaptadors de dades corresponents. Així tenim un mecanisme que fa de "plataforma d'estandardització de dades", una capa de dades homogènies entre tots els subsistemes de planta [21]. Per això, a nivell del ERP, els importadors externs del capítol 4.2.4 tradueixen la informació entre la base de dades de planta (estàndard) cap al format propi del ERP. D'aquesta manera, el model teòric de

planta permet una major reutilització de les components. Per exemple, el "NestingImporter" rebrà el missatge de conversió del "Translator" corresponent i importarà la informació del document estàndard que trobarà a la base de dades de planta. En cas de canviar el sistema ERP només caldria retocar en "NestingImporter", mantenint en funcionament els "Translators" i "watchers" corresponents. Per altra banda, es poden afegir nous mòduls CAD i només s'han d'afegir els "Translators" a mida. A més, la informació estàndard passarà a ser disponible per a altres subsistemes que no en podien disposar amb el model antic. A nivell d'implementació real, es pot fer aqueta estandarditzación representant els documents en format XML, amb esquemes XSD estandarditzats [77]. Per exemple, es poden utilitzar alguns dels estàndards EDI [29] per a la informació de dades de gestió, STEP [29] per a la representació del model del producte, etc. A més, si s'incorpora la utilització d'una ontologia a la representació de la informació, aquestes dades estandarditzades només incorporaran conceptes que estiguin al domini d'aplicació de la ontologia, cosa que permetrà augmentar les capacitats deliberatives automàtiques de les entitats que s'incorporin al model en un futur, i els guanys en consistència de la informació també seran evidents (tots els conceptes de la base de dades tindran el mateix "significat" per a tots el mòduls que els utilitzin).

4.4 Control dels magatzems

El model funcional dels magatzems resumeix els requeriments d'aquestes entitats físiques per a la seva integració amb el sistema de planta. Amb un model tan simple com el de la Figura 30 es poden incloure els magatzems automatitzats parcialment, completament robotitzats o el clàssic magatzem "manual". Com que es tracta d'entitats físiques, el refinament del model amb propòsit d'optimització o control directe de la part física implicaria la modelització d'entitats separades en funció del tipus de magatzem, i centrada en aspectes de l'arquitectura concreta del tipus de magatzem.

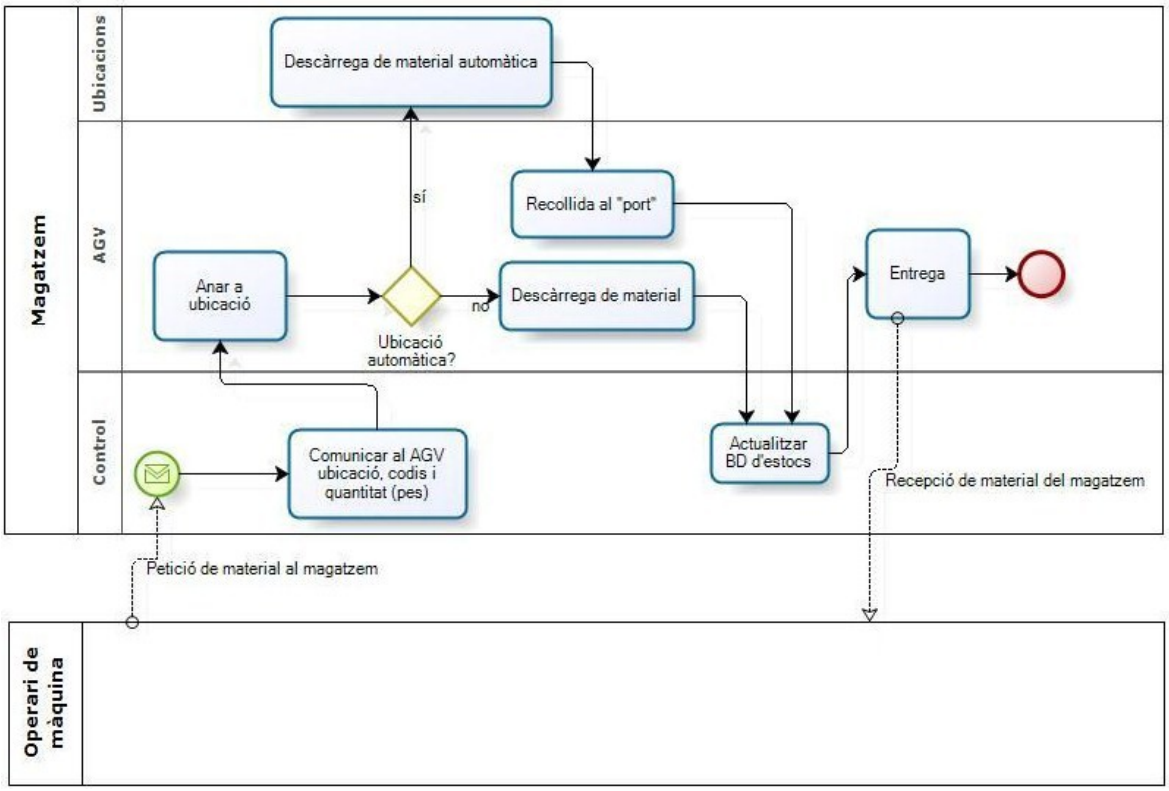


Figura 30. Model de magatzem genèric

En el cas de la planta real que es tracta en aquest treball hi ha 3 tipus de magatzems:

- Magatzems verticals automàtics: es tracta de contenidors en forma de prisma rectangular. Normalment s'utilitzen per emmagatzemar peces semi fabricades, utillatges i "cargoleria". El control es fa mitjançant una consola a la seva base, disposant els elements sobre una safata. Les ubicacions dins el magatzems estan totalment automatitzades així com el control del seu contingut (estoc disponible de cada element).
- Magatzem no automatitzat: aquest és el clàssic magatzem on un conjunt de prestatges fan d'ubicacions per classificar el tipus d'elements que contenen. Normalment s'hi guarden productes semi fabricats i els productes acabats pendents d'entrega. No hi ha automatització, la identificació de les ubicacions i elements es fa mitjançant codis de barres, i el transport/recollida el fan operaris humans mitjançant carretons.
- Magatzem automàtic de matèria primera: aquest magatzem completament automatitzat emmagatzema la matèria primera, en aquest cas planxes de diversos tipus de metalls i dimensions. Aquest s'estén al llarg d'una nau sencera de la planta, amb les màquines que processen la matèria primera al seu costat. En un extrem hi ha el control d'entrada de material i d'entrega, mitjançant una consola. Un vehicle automatitzat, bàsicament un pòrtic que es desplaça sobre uns rails, recull el material sol·licitat o l'entrega. L'entrega de material es fa directament a la màquina que ho

sol·licita, si és una de les que estan disposades al costat dels rails per on es desplaça el vehicle automatitzat del magatzem.

El model de la Figura 30 permet representar aquests 3 tipus de magatzems dividint la seva funcionalitat en tres rols diferenciats. Tot i que són 3 funcionalitats de la entitat “magatzem” es poden implementar físicament separats, tot i que fortament lligats. El propòsit és representar els tres aspectes bàsics del comportament dels magatzems a nivell de planta: el seu control, el transport del material i la col·locació dels elements dins el magatzem. Tots 3 aspectes poden ser implementats mitjançant mòduls específics, agents físics, controlats per agents software, etc. Així i tot, la part de transport de material s’anomena “AGV” al model teòric perquè en una planta totalment integrada i agentificada el model preveu que el moviment de material entre i des dels magatzems el facin vehicles autònoms. La funcionalitat que s’ha agrupat a l’esquema sota el nom “ubicacions” fa referència a la capacitat dels magatzems per situar i organitzar els elements dintre seu.

Com en esquemes anteriors, aquí es mostra sense detall l’entitat separada “operari de màquina”, que es relaciona amb el model de magatzem mitjançant peticions i rebent material. A continuació es detalla el seu esquema.

4.5 Operari de màquina

L’entitat encarregada de supervisar i controlar el treball de les màquines eina s’anomena “operari de màquina” dins el model teòric perquè a l’esquema tradicional de planta les seves tasques les realitzen operaris humans. Bàsicament s’encarrega de dos tipus de procediments principals, que es mostren a l’esquema de la Figura 31: gestió del material, és a dir, preparar i posar a la màquina la matèria primera o els elements semi fabricats (per això es mostra el “diàleg” amb el magatzem), i gestió de la informació de producció de la màquina. Aquest darrer rol consisteix en fer d’intermediari entre el sistema ERP, i la màquina, donant el vist i plau a la ordre de fabricació amb que ha de treballar la màquina, carregant-li els programes CN, o, simplement, fent d’operador manual de màquina. En aquest rol d’enllaç amb el ERP, o de gestió de la informació, l’operari també s’encarregarà de passar informació del nombre d’elements processats i del temps de treball al ERP. Aquest és un altre exemple de processos del model teòric de planta candidats a ser automatitzats o agentificats. A la planta real, la gestió del material i el control de màquina es fa mitjançant robots en alguns casos concrets de màquina.

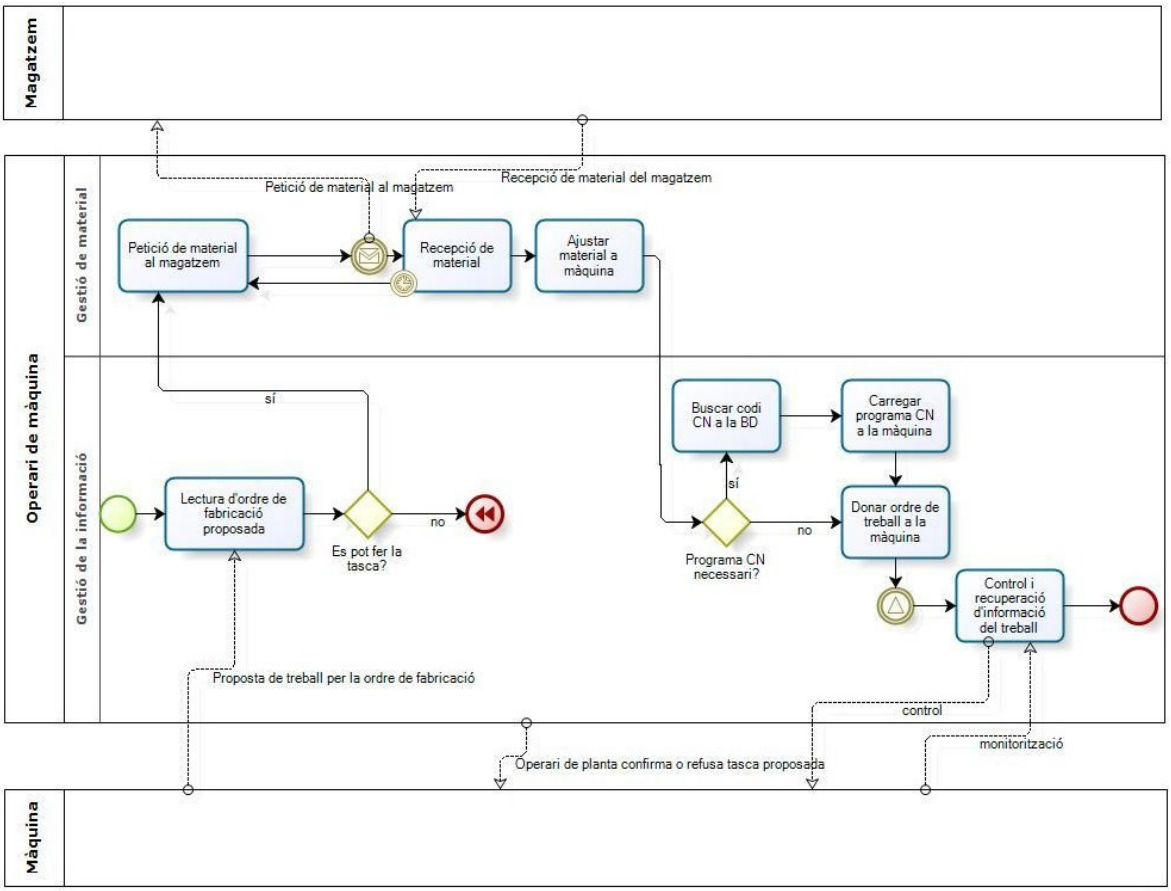


Figura 31. Model de processos de l'operari de màquina

4.6 Màquina

La característica més important que s'observa a l'esquema de la Figura 32 és que les màquines del model teòric de planta tenen un comportament actiu en quant a la gestió dels treballs que accepten per processar. Això es reflexa a l'esquema mostrant les màquines com a entitats amb dos comportaments o rols ben diferenciats. Per una banda la part de treball, en la qual reben les ordres de l'operari que dóna el vist i plau al començament dels treballs, controla i recull dades del treball. Per l'altra banda es comporten com a cues de treballs on les ordres de fabricació es posen en espera de ser processades. En aquest cas, la màquina té accés al ERP, comprova els requeriments del treball a fer (per exemple si es pot processar abans de la data objectiu del treball dins del pla de processos de la ordre de fabricació, o bé mirant si es compleixen els requeriments de dependències entre processos al pla de la ordre de fabricació), i el posa a la cua de treballs en funció dels seus requeriments o bé el refusa. La cua de la màquina també és la encarregada de proposar la propera feina a l'operari de màquina. Aquest comportament actiu fa que les màquines siguin sistemes candidats a ser agentificats, amb la dificultat afegida d'haver de controlar el sistema físic.

Un altre aspecte que no s'ha reflectit en aquest primer model de planta, però que es pot detallar en properes versions és el fet de preveure una capacitat de diàleg entre les màquines. Aquestes podrien, per exemple, "passar-se" feines en cas d'avaria, saturació o possibles imprevistos.

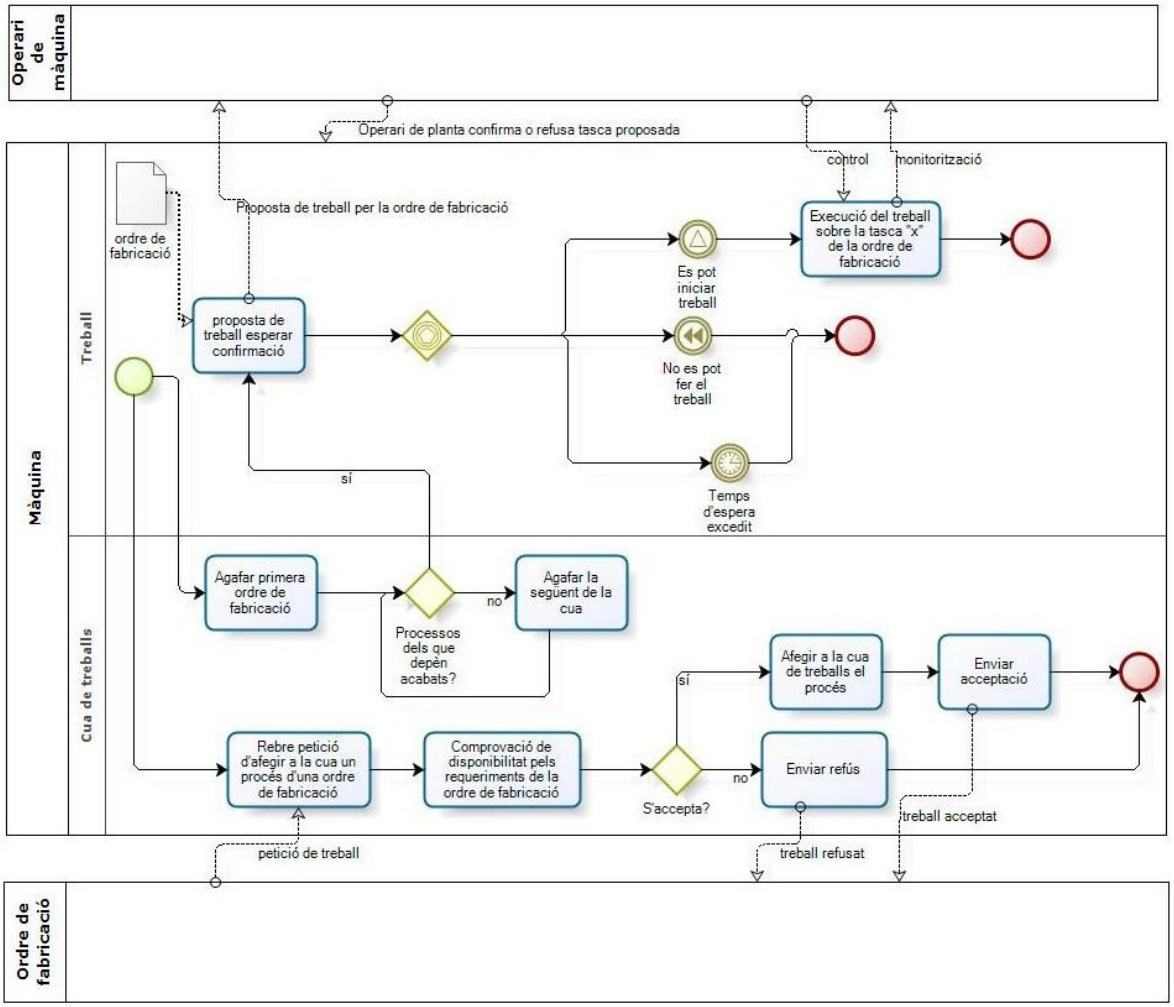


Figura 32. Model de màquina amb cua de treballs associada

El darrer aspecte notori del model de màquina és el fet de l'aparició de les ordres de fabricació com a entitats separades i també actives, encarregades de negociar amb les màquines la seva entrada a les cues de treballs d'aquestes, dins del model teòric. L'esquema detallat d'aquestes es mostra a continuació.

4.7 Ordre de fabricació

Com s'ha vist al capítol 4.2 el model de processos de les ordres de fabricació conté el seu pla de fabricació. Dins del model teòric aquestes es poden veure com a entitats separades, "llançades" pel ERP a planta, i s'encarreguen de negociar amb les màquines l'execució de les tasques del seu pla de fabricació. A la Figura 33 es veu com l'ordre de fabricació demana a les màquines disponibles per cada procés si aquest es pot afegir a la seva cua de feines. També es mostra de forma

resumida el cicle de vida d'una ordre de fabricació: enviar els processos del seu pla a les màquines, esperar que les màquines treballin, passar a oficina pel control de qualitat, i tornar a planta si no passa el control de qualitat.

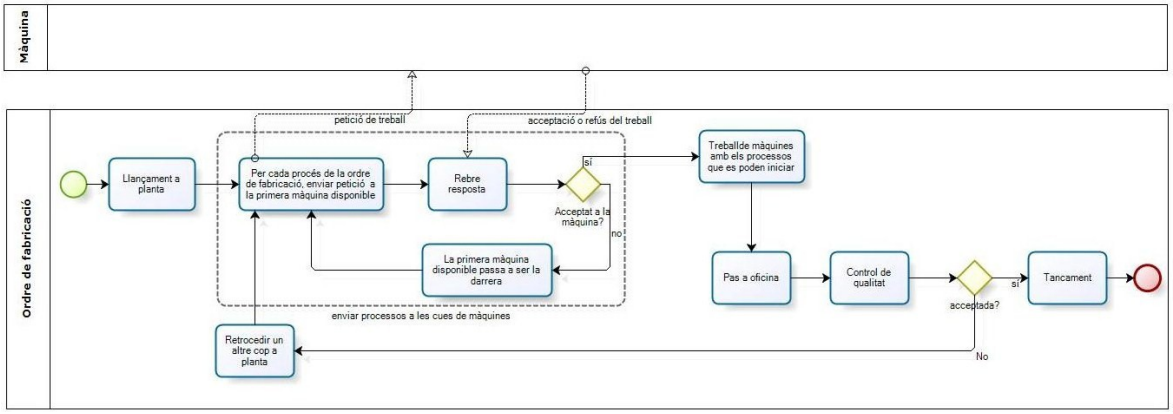


Figura 33. Ordre de fabricació

La conversió de les ordres de fabricació en les entitats autònomes que mostra el model teòric requereix una anàlisi exhaustiva abans de la seva implementació a planta real. Això es deu a que, en un exemple de planta relativament petita com la de l'exemple, solen haver-hi unes 4000 ordres de fabricació actives a planta, cadascuna amb el seu pla de fabricació de no menys de 5 processos (per exemple tallar, punxonar, doblar, soldar i pintar). És per això que cal tractar amb el problema d'implementar o incorporar tecnologia que suporti l'existència d'unes 8000 entitats software actives al sistema, ja siguin *threads*, agents o processos.

5. Implantació parcial del model teòric

5.1 Introducció

El model teòric de planta, que es mostra al capítol 4, és el primer pas a l'aplicació de la solució proposada pel cas de la planta real. Com que la solució consisteix en una transformació progressiva del sistema la implantació d'aquest model es fa de manera parcial. En aquest capítol es mostra la primera implantació parcial del model a l'exemple real, tenint en compte que en aquest treball només s'ha obtingut una primera versió del model teòric de planta sense incloure el model sencer d'empresa. Es tracta d'un treball incipient pel que fa a la part real implantada, si s'observen el model sencer i l'esquema de planta real. A més, en el cas real s'ha hagut de fer la implementació de mòduls sota restriccions i limitacions concretes de tecnologia, temps d'implantació i organització d'empresa, més enllà de les restriccions mostrades al capítol 1.4 . Així i tot, el sol fet d'haver obtingut el model teòric, una representació estandarditzada de la informació i una mínima implementació ja permeten obtenir alguns dels avantatges comentats al capítol 1.6.

A continuació es fa una descripció de la implantació en termes del model de planta, és a dir, s'assenyalen algunes parts crítiques del model i es justifica la seva transformació. Posteriorment es busquen les equivalències entre el model i els elements reals a planta pel que fa a la part que es vol transformar. Finalment s'explica amb detall la part real implementada i s'exposen les restriccions i condicionants de treball que poden fer que en aquest cas no sigui possible seguir de forma estricta la metodologia proposada al capítol 1.5.

5.2 Implantació a partir del model teòric

Com que el model es centra en la planta, la majoria de processos giren al voltant de les ordres de fabricació (OFs), el seu "cicle de vida". En primer lloc, les tasques d'oficina tècnica consumeixen bona part del temps en la creació i optimització dels models del producte (capítol 4.2). Per això la part del model on es tracta la generació de les ordres de fabricació és el punt on interactuen un major nombre de mòduls software heterogenis al voltant del sistema ERP. Aquesta part del model inclou des de la creació del model de producte fins al llançament de les OFs a planta. La integració d'informació s'assigna a entitats que permeten incorporar nous mòduls CAD/CAM amb un mínim esforç (capítol 4.3): "Watchers", "Translators" i "Importers". A més de permetre incorporar o substituir mòduls software, també substitueixen processos assignats a personal humà dins

del ERP, i tasques iteratives que poden produir redundància de dades (per exemple la importació de BOM o dades de programes i Nestings).

Per altra banda, el model també mostra una part crítica a l'entitat que representa les màquines (Figura 32). En concret el paper de la màquina com a cua de treballs requereix una certa negociació amb les OF i, el més crític, requereix una política d'ordenació de la cua que permeti assolir els objectius dels plans de fabricació de cada OF. Per tant, el comportament de les cues de màquines i, si s'amplia, la negociació amb les OF definiran en bona part el rendiment del procés productiu.

Finalment l'operari de màquina és l'entitat que fa de pont entre les màquines, els sistemes d'informació i la gestió de materials. L'automatització o millora dels processos assignats a aquesta part del model eliminarien un altre coll d'ampolla (capítol 4.5).

En qualsevol cas, tots aquests punts crítics treballen sobre la informació que conté el mòdul de producció dintre del sistema ERP, les entitats "ordre de fabricació" al model. Les OF es relacionen amb les cues de les màquines, els operaris de planta decideixen començar els treballs avaluant el pla de fabricació de les OF, també recuperen els programes CN i els carreguen en funció de la informació continguda a aquest pla.

Per això, degut a la naturalesa heterogènia de les entitats que recuperen i manipulen aquesta informació es fa necessari, en primer lloc, facilitar aquest intercanvi d'informació i eliminar redundàncies de processos i dades. En el cas d'aquest treball, la primera experiència d'implantació és un treball preliminar, molt restringit, consistent en la implantació de la part del model que tracta amb la integració de mòduls software heterogenis (capítol 4.3). D'aquesta manera s'intenten assolir dos objectius: en primer lloc un primer pas cap a la existència d'un sistema d'informació estàndard i global a planta (canal de dades estandarditzades) i, en segon lloc, agilitzar la gestió de dades tècniques d'oficina a l'hora d'incorporar nous models de producte i llançar-los a planta. Així en aquesta primera etapa no s'han de tractar problemes propis de sistemes físics, hardware i temps real. Però igualment es poden observar els avantatges, quant a rendiment i modularitat, que ens aporta un mínim treball sobre un model de solució vàlid per al problema.

5.3 Correspondència amb el model teòric

A continuació es mostren les equivalències entre mòduls i components existents a la planta real, i les components del model funcional mostrades al capítol 4.3. Això es fa com a pas previ a la implantació d'aquesta part del model, integració d'elements software heterogenis i, en concret, la integració de la informació dels mòduls de nesting i CAD amb el sistema ERP. Les mancances seran punts a implementar i les correspondències poden suposar restriccions de treball (si es decideix conservar-les).

5.3.1 Sistemes externs de CAD/CAM

Segons el model, aquests mòduls proporcionen l'estructura de producte, informació dels elements de les BOMs, així com els programes CN per a les màquines. Pel que fa als mòduls CAM, encarregats del nesting, en aquesta primera fase s'implantarà el model per les màquines de tallar i punxonar, que són els processos crítics quant a despesa de matèria primera. La implantació del model als mòduls CAM de torns i plegadores es farà en fases posteriors.

En el cas de la planta real tenim els següents sistemes llegats que es corresponen amb el "CADModule" i "NestingModule":

- CAD: es disposa de diverses estacions de treball (entre 8 i 12), gairebé totes amb PRO/ENGINEER® (per al disseny 3D i algunes estacions (unes 4) amb AutoCAD LT®. Tots ells permeten l'exportació del model geomètric cap al format DXF i la programació de macros i funcions per exportar les llistes de materials. En el cas de PRO/ENGINEER es disposa de macros que generen informes en fitxers de text amb un resum de la llista de materials i les característiques de cada element d'aquesta. Aquests fitxers es generen automàticament.
- CAM: centrant-nos en els mòduls de nesting dels sistemes CAM, es treballa bàsicament amb JET-CAM®. Es tracta d'un software molt utilitzat a la indústria del metall, i suporta la majoria de màquines de punxonar i tallar. Es pot configurar perquè, cada vegada que es genera codi CN, exporti un fitxer de text amb la informació de les peces contingudes al nesting, així com dades de fabricació i utilització de planxa. Així i tot, també es disposa del software BYSOFT®, específic per a màquines Bystronic® però la seva utilització es redueix a la optimització de materials i generació de codi CN per 3 màquines làser concretes. Tot i que està gairebé en desús a planta, i és una versió antiga, la informació de les peces i programes és accessible directament a la seva base de dades.

5.3.2 LegacyWatcher i Translator

Tornant al model de la Figura 29, aquestes dues entitats són les encarregades de detectar actualitzacions de documents als sistemes externs i de fer-ne la traducció cap a una representació estàndard. Actualment no estan implementades a la planta real, per això la importació d'informació des dels mòduls CAD/CAM la realitzen els operaris d'oficina tècnica. Es fa necessària la implementació d'aquesta part del model per als mòduls software externs CAD/CAM. Els mòduls que es volen integrar generen automàticament fitxers de text (amb un format propi), que contenen la informació especificada pels tècnics d'oficina. L'únic mòdul que no exporta explícitament la informació té la seva base de dades accessible. Per tant, la opció més simple, a priori, pot ser considerar que l'actualització del contingut d'algun sistema de fitxers és el senyal que indica al "watcher" que cal fer una traducció (veure Figura 28). Amb això es poden utilitzar diverses instàncies del LegacyWatcher per controlar l'actualització dels documents a

diversos equips físics. Per als Translators és necessària una anàlisi del format i tipus d'informació exportada.

5.3.3 BomImporter i NestingImporter

Aquestes dues entitats, encarregades d'enllaçar la informació estandarditzada amb la base de dades ERP, també s'han d'implementar. En aquest cas, però, la tecnologia per a la gestió de dades estàndard serà la mateixa que als Translator, i no cal una anàlisi/reinterpretació d'aquesta informació. Per l'altra banda, l'accés a la informació del sistema ERP, i als seus mòduls, es fa mitjançant llibreries i tecnologia específiques proporcionades pels enginyers propietaris del sistema. Això suposa unes restriccions tecnològiques d'implementació per aquest enllaç.

El sistema ERP està format per un conjunt de mòduls que s'executen de forma distribuïda a la planta, amb una base de dades propietària centralitzada. Els desenvolupadors proporcionen llibreries de classes implementades amb tecnologia Microsoft .NET Framework® [52]. Aquestes permeten crear, recuperar i modificar elements del sistema ERP. Per exemple, la llibreria de producció inclou classes per manipular objectes "Machine", "Process", "Order", etc. També s'inclouen rutines d'actualització (com per exemple per crear ordres de fabricació a partir de BOMs) i classes per comunicar-se amb els mòduls del sistema que s'executen de forma distribuïda, mitjançant la tecnologia .NET Remoting®. Per tant, com que en aquesta és una primera implantació molt parcial, gairebé un treball preliminar per veure si el model de solució és vàlid, aquestes llibreries suposen un condicionant tecnològic important. Es començarà implementant aquests primers mòduls d'integració amb la tecnologia .NET.

5.4 Implementació a la planta real

La revisió de les correspondències entre el model teòric i el model existent a planta, han mostrat que cal fer la implementació del LegacyWatcher, els Translators i els Importers. A més, elements ja existents a planta, i la tecnologia disponible a l'entorn real han portat a introduir alguns condicionants per a la implementació real. Anem ara a veure, com s'ha fet la implementació d'aquestes noves entitats, la seva implantació a planta i els resultats obtinguts.

En primer lloc, cal tenir present que la implementació real mostrada aquí només ha estat un primer pas, molt preliminar, per mostrar la validesa de la metodologia en un cas concret i encoratjar un treball més profund i extens seguint la metodologia proposada al capítol 1.5. Per això primerament es fa un repàs als condicionants per al desenvolupament i implantació a la planta real.

5.4.1 Condicionants tecnològics

La implementació s'ha fet en una planta real, d'una empresa real, on el temps i recursos són limitats. A més, algunes tasques requereixen la coordinació amb els

enginyers de l'empresa a la planta, i aquests també desenvolupen i mantenen el sistema. Per això el primer requeriment ha estat la utilització, fins on sigui possible, dels seus entorns de desenvolupament i llibreries. A planta, exceptuant alguna consola d'accés a les controladores de control numèric, els equips treballen sota diverses versions de Microsoft Windows®. A més, es disposa de l'entorn de desenvolupament Microsoft Visual Studio 2005 amb el que els enginyers de planta desenvolupen alguns mòduls ad-hoc, llibreries d'accés a altres sistemes software i serveis per a la intranet. Per això les terminals i servidors disposen de la infraestructura .NET Framework®.

En aquest cas, doncs, tot i que s'han fet proves de laboratori i és una tecnologia més adient, no s'utilitza un entorn de desenvolupament (amb plataforma) multiagent, com per exemple JADE. A més, en el procés d'anàlisi dels mòduls CAD/CAM en funcionament, s'ha fet necessari el diàleg i coordinació amb el personal d'oficina tècnica. Amb tot això, el temps requerit per aquesta coordinació, anàlisi i documentació, porta a deixar per una propera fase el seguiment de la metodologia formal proposada al capítol 1.5.

5.4.2 Model de comunicacions

El requeriment d'un desenvolupament per a .NET Framework®, juntament amb la interfície d'accés al ERP també impedeixen implantar una capa de comunicacions per a agents estàndard. Això vol dir que no es disposa d'una capa de comunicacions que proporcioni mecanismes (en forma de llibreries de classes, per exemple) per encapsular el contingut dels missatges dintre d'un format d'ACL estàndard (per exemple FIPA-ACL), ni eines per interpretar la intencionalitat d'aquests missatges o, a més, facilitar la programació de protocols d'interacció. De totes formes, l'entorn d'aplicació totalment acotat, a més de tractar-se de la implementació d'una part molt petita del model, permeten fer aquesta primera implementació sense la part estàndard d'un ACL. Com que el model de comunicacions existents es basa més en la crida a mètodes d'objectes distribuïts per la xarxa que en el pas de missatges explícits, els mateixos mètodes i objectes contenidors d'aquests representaran la intencionalitat dels missatges. El que sí que s'ha fet és estructurar de forma estàndard el tipus d'informació que es passen els mòduls entre ells, és a dir, la part de contingut dels missatges. D'aquesta manera, quan es pugui implantar una capa de comunicacions entre agents amb suport per a ACL estàndard, el codi per generar i processar aquests continguts serà igualment vàlid, només caldrà embolcallar aquesta informació dintre del camp de contingut dels missatges, i implementar la part de gestió dels protocols d'interacció.

Així doncs, seguint els requeriments d'implementació, la comunicació entre els elements implementats es fa mitjançant la tecnologia .NET Remoting® [51]. Remoting, dins de Microsoft Framework, permet distribuir aplicacions i fa transparent el mecanisme de comunicació entre processos, és la tecnologia successora del model DCOM. Facilita la interacció entre processos dins d'un

mateix equip o entre equips distribuïts per la xarxa, i ho fa separant aquesta interacció del mecanisme específic de comunicacions o del tipus d'aplicació contenidora dels processos (per exemple, una aplicació de Windows es pot comunicar amb un servei web).

Resumint, els principals conceptes de Remoting són:

- Comunicacions transparents: perquè els mètodes d'una classe es puguin cridar de forma remota, aquesta ha de derivar de la classe *MarshalByRefObject.* Això és independent al tipus de canal de comunicacions utilitzat. D'aquesta manera, quan es crea un objecte remot dins d'un equip client, Remoting crea un objecte proxy amb els mateixos mètodes i propietats que l'objecte remot, i s'encarrega de redirigir les crides a aquest objecte cap al procés que s'executa al servidor remot.
- Utilització de canals: els canals de transport representen el conjunt de tecnologies subjacents necessàries per obrir la connexió de xarxa i enviar un paquet de dades. Per això es pot veure un canal com un objecte capaç d'enviar i/o rebre informació. Es proporcionen canals predeterminats que permeten escoltar i enviar missatges segons protocols estàndard encara que a l'altre extrem no hi hagi el Microsoft Framework instal·lat, per exemple *TCPChannel* i *HttpChannel*. El programador té la llibertat d'utilitzar qualsevol dels canals registrats al servidor per comunicar-se amb els objectes remots que aquest conté.
- Aplicació contenidora i fitxers de configuració: al servidor, es fa necessari un procés que escolti per canals concrets i redirigeixi les peticions a les llibreries que proporcionen les classes d'ús remot concretes. Per aconseguir això s'utilitzen fitxers de configuració on s'especifica, entre d'altres coses, la llibreria (*assembly*) , la classe remota, els canals utilitzats, i l'identificador universal de l'objecte remot(URI). L'aplicació contenidora (*listener*) només ha de carregar aquests fitxers de configuració i Remoting fa la resta.

  A l'exemple de la Figura 34 es mostra el fitxer de configuració per a la classe *RemotableType* continguda a la llibreria del mateix nom. En aquest cas només s'utilitza un canal http. Perquè un procés servidor pugui utilitzar aquest fitxer i redirigir les peticions cap a la llibreria concreta només ha d'utilitzar Remoting i carregar el fitxer de configuració. Per exemple, en llenguatge C# només caldria fer el següent:
  RemotingConfiguration.Configure("Listener.exe.config");

```
<configuration>
   <system.runtime.remoting>
      <application>
         <service>
            <wellknown
               mode="Singleton"
               type="RemotableType, RemotableType"
               objectUri="RemotableType.rem"
            />
         </service>
         <channels>
            <channel ref="http" port="8989"/>
         </channels>
      </application>
   </system.runtime.remoting>
</configuration>
```

Figura 34. Configuració de classe remota [51]

On “Listener.exe.config” és el nom del fitxer de configuració. En el cas de les aplicacions client també s’utilitzen fitxers de configuració. El fitxer de configuració de la Figura 35 permet a un client utilitzar la classe *RemotableType*. Es pot observar com el camp “url” permet localitzar el servidor remot (a l’exemple, però, es fa sobre el mateix equip) utilitzant els paràmetres de configuració que s’han mostrat a la configuració de servidor (Figura 34).

```
<configuration>
   <system.runtime.remoting>
      <application>
         <client>
            <wellknown
               type="RemotableType, RemotableType"
               url="http://localhost:8989/RemotableType.rem"
            />
         </client>
      </application>
   </system.runtime.remoting>
</configuration>
```

Figura 35. Fitxer de configuració per aplicació client [51]

5.4.3 Integració de la informació

La part del model teòric implementada en aquest treball requereix d'una representació estandarditzada de la informació. D'aquesta manera es redueix el nombre d'agents importadors necessaris i es fa una primera aproximació a la solució del problema de la integració de dades heterogènies. Aquesta representació només és un primer pas perquè el problema de la heterogeneïtat es pot tractar a diversos nivells, alguns d'ells encara formen part de la recerca i s'escapen a l'àmbit d'aquest treball. Sheth [71], per exemple, identifica 4 nivells d'heterogeneïtat: de sistema (hardware i sistemes operatius incompatibles), estructural (diferents models de dades, per exemple bases de dades diferents), sintàctica (diferents llenguatges i representació de dades) i semàntica (el significat de la informació intercanviada).

En el cas de la planta exemple, gairebé no existeix a nivell de sistema (excepte per algunes terminals hardware específiques o algun dispositiu mòbil), en qualsevol cas hi ha diverses tecnologies i middleware que permeten tractar aquest tipus d'heterogeneïtat. Aquí ja s'ha mostrat .NET Remoting com a tecnologia per a processos distribuïts a planta. Pel que fa a nivell estructural, la utilització de models estàndard de dades també faciliten la integració, en aquest cas SQL. El problema de la integració a nivell semàntic és el més complex, els problemes associats la interoperabilitat semàntica es deuen a que la interpretació de la informació depèn del context d'aplicació [17]. Per això es treballa en la utilització d'ontologies com a sistema de modelat del context per tractar aquest tipus de problema [21][53]. Aquest és un problema que requereix un treball més profund i planteja diversos problemes, com la utilització del llenguatge de modelat de la informació, expressivitat de la semàntica etc. Per això en aquesta implementació no s'ha treballat en la interoperabilitat a nivell semàntic.

Així doncs, aquí només s'ha tractat la integració a nivell sintàctic, considerant, a més, que no existia un espai de dades estandarditzades global a tots els sistemes de planta, i que, un cop creat aquest, s'utilitza de moment només a nivell intern i sempre en el mateix context. Per a aquesta integració de la informació a nivell sintàctic s'ha utilitzat XML [77] com a estàndard de representació. XML és àmpliament utilitzat per a intercanviar d'informació entre sistemes heterogenis. Amb XML els documents estan estructurats i etiquetats, d'aquesta manera contenen les dades i la informació necessària per extreure-les i interpretar-les. A més, la utilització dels *document type definition (DTD)* i dels esquemes [78] XSD permet afegir restriccions gramaticals més fortes. Els esquemes, a més de les possibilitats dels DTD, proporcionen més opcions per definir *namespaces*, més tipus de dades bàsiques, combinació de documents diversos. A més, els esquemes són documents XML en si mateixos, el que els fa més tractables[17]. Per tant, un document XML serà vàlid si té un esquema associat i compleix les restriccions gramaticals que aquest imposa.

Un altre avantatge és que actualment la majoria de plataformes de desenvolupament suporten XML i proporcionen eines per processar els

documents i verificar documents amb els esquemes i DTD. Per això, tot i que en aquest treball s'han utilitzat les eines que proporciona el Microsoft Visual Studio 2005, en properes ampliacions o substitucions del middleware de planta es podran conservar els esquemes i documents XML. La transmissió (intercanvi) de dades electròniques (EDI) és un altre motiu per a la utilització de XML. Actualment el sistema ERP permet intercanviar electrònicament alguns tipus de documents mitjançant estàndards tradicionals com UN/EDIFACT, que no són representacions XML. Però els sistemes ERP estan tendint cap a la incorporació d'estàndards XML per a intercanviar de documents. Per exemple, UBL [61] (continuació dels treballs fets amb ebXML[60]), proporciona esquemes pels documents de negoci bàsics ("PackingList", "ApplicationResponse", "BillOfLading", etc.), descripcions acurades dels models de processos en que intervenen aquests documents XML i diverses especificacions complementàries (per exemple una per documents ABN en forma binària).

Tenint en compte que a planta l'XML només s'utilitzava de forma temporal, en algunes aplicacions, per exportar informació al clients i proveïdors (i només en comptades ocasions), el primer pas ha estat establir esquemes per als tipus de documents que es vol integrar en aquesta primera fase. Amb aquest primer pas s'estandarditza a nivell de planta el vocabulari utilitzat per fer referència als conceptes de producció bàsics i s'estructura la seva representació. A més, si en properes ampliacions es vol tractar el problema de la heterogeneïtat a nivell semàntic, es pot aprofitar aquesta estructura de conceptes a l'hora d'implementar o adaptar una ontologia per a aquest domini d'aplicació. Tot i que aquí no es tracta aquest problema, els tipus d'ontologies més simples i no formals es basen en una classificació taxonòmica dels conceptes del domini [21].

Així doncs, en primer lloc s'han implementat esquemes per a la representació en XML de la informació bàsica dels nestings i les estructures de les BOM. Els Translators han de transformar la informació que troben en format propi cap a documents XML vàlids segons aquests esquemes. A continuació es mostra, de forma resumida, una part de l'esquema de nestings per veure com es defineix l'estructura i els valors del document.

La Figura 36 conté la definició bàsica de l'estructura d'un programa. Els símbols "+" a l'esquerra dels nodes etiquetats indiquen que s'ha ocultat part de l'estructura per simplicitat. Aquest exemple serveix per veure les característiques bàsiques dels esquemes. Es permet la definició d'elements niats (elements dintre d'elements) i d'estructures complexes. En aquest cas els elements "StdDataProgram" contenen elements "StdDataControl", "StdDataGeneral", etc. A més, s'especifiquen diverses formes d'estructurar aquests elements niats com, per exemple, seqüències d'elements que han d'aparèixer en aquest ordre al document, o bé conjunts d'elements a escollir. Els atributs associats als nodes d'aquests elements amplien les restriccions gramaticals. Tornant a la Figura 36, un programa estarà format per les dades de control "StdDataControl", les dades generals "StdDataGeneral", potser alguns documents associats (els atributs "minOccurs" i "maxOccurs" indiquen les quantitats mínima i màxima), després

apareixen les estimacions de treball "StdDataEstimation" i, en darrer lloc, un conjunt d'utillatges de màquina "stdPunchingTools". En aquest darrer cas poden aparèixer de 0 a 99999 utillatges a escollir entre un conjunt de "StdDataPunchingTools". En el cas del "StdDataDocument" no s'especifica la definició dintre del "StdProgram" perquè és una referència a un tipus d'element definit fora de la branca de l'arbre (reutilitzable), un altre dels avantatges dels esquemes.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://tempuri.org/XSDSchema1.xsd" xmlns:mstns="http://t
  <xs:element name="StdDataProgram">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="StdDataControl">
        </xs:element>
        <xs:element name="StdDataGeneral">
        </xs:element>
        <xs:element ref="StdDataDocument" minOccurs="0" maxOccurs="1000"/>
        <xs:element name="StdDataEstimation" minOccurs="0">
        </xs:element>
        <xs:choice minOccurs="0" maxOccurs="99999">
          <xs:element name="StdDataPunchingTools">
          </xs:element>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="StdDataDocument">
  </xs:element>
  <xs:complexType name="StdDataSection">
  </xs:complexType>
</xs:schema>
```

Figura 36. Esquema per definir un document de nesting en XML

En alguns casos, a més dels elements que composen el document, pot ser necessari associar informació addicional als elements del document XML, per això l'esquema permet definir els atributs associats als elements del document, amb totes les restriccions gramaticals necessàries (tipus de dades, si aquest és obligatori, etc.). La Figura 37 conté la definició dels utillatges de les màquines de punxonar (a la Figura 36 estava oculta). Aquesta part del document haurà de contenir l'element d'utillatge concret (a escollir entre circular, rectangular, etc.), notes descriptives i un nom d'utillatge. A diferència de la figura anterior, aquí s'han definit alguns atributs que s'hauran d'omplir al document per aportar tota la informació necessària (paràmetres com la tolerància o l'angle de l' utillatge a la màquina).

```
<xs:complexType>
  <xs:sequence>
    <xs:choice>
      <xs:element name="StdDataCircular">
      </xs:element>
      <xs:element name="StdDataRectangular">
      </xs:element>
      <xs:element name="StdDataSquare">
      </xs:element>
      <xs:element name="StdDataColiso">
      </xs:element>
      <xs:element name="StdDataSpecial">
      </xs:element>
      <xs:element name="StdDataEllipse">
      </xs:element>
    </xs:choice>
    <xs:element name="notes" type="xs:string"/>
    <xs:element name="toolType" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="tolerance" type="xs:decimal" use="required"/>
  <xs:attribute name="station" type="xs:decimal" use="required"/>
  <xs:attribute name="angle" type="xs:decimal" use="required"/>
  <xs:attribute name="multi-tool" type="xs:decimal" use="required"/>
  <xs:attribute name="heigh" type="xs:decimal" use="required"/>
</xs:complexType>
```

Figura 37. Definició de l'element "StdDataPunchingTools" amb atributs específics

L'estructura i atributs de l'esquema de nestings també es pot visualitzar gràficament, de forma molt simplificada, per comentar-ne els elements més importants:

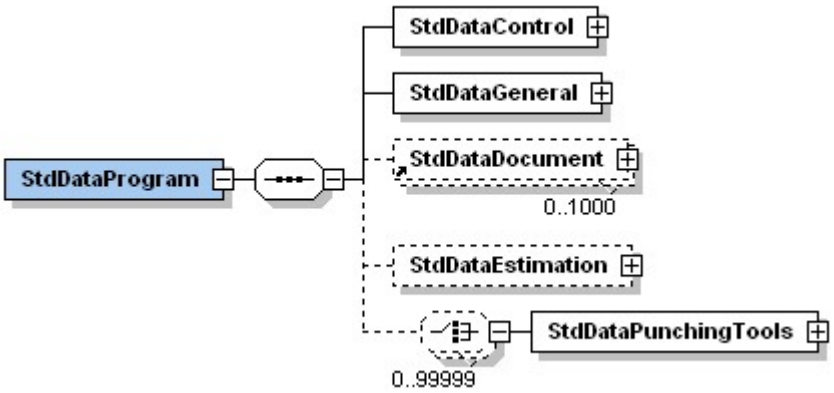


Figura 38. Representació gràfica de l'esquema de programa de nestings

- StdDataProgram (Figura 38): l'element del document que representa el programa de nestings. És la representació gràfica de la definició de la Figura 36.
- StdDataControl: un element bàsicament composat per atributs de control del document XML, com , per exemple, data de conversió, data de creació, mòdul que l'ha generat, tipus de document original, etc.

- StdDataGeneral (Figura 39): aquesta part del document ha de contenir la informació general del programa. Els elements més importants són la màquina, el material utilitzat i el nesting (peces niades).

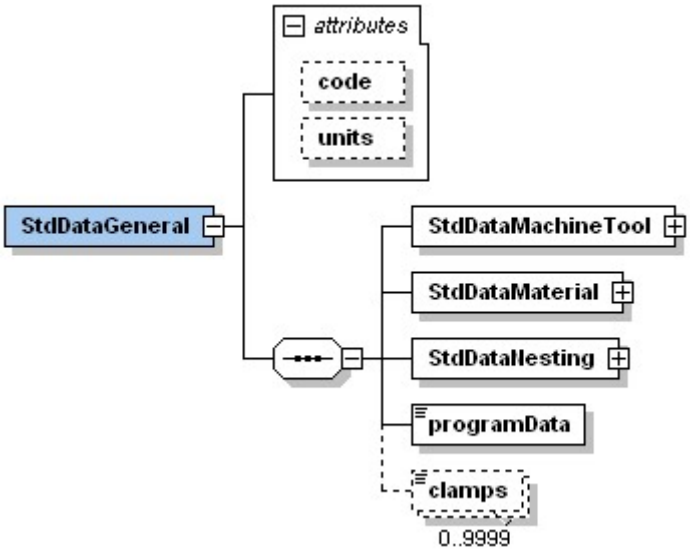


Figura 39. Definició de les dades generals del programa

- StdDataMaterial (Figura 40): aquest element és la definició de la informació relacionada amb el material sobre el que treballa el programa a la màquina. Aquí només es mostra l'element "StdDataSheet" (planxa de metall), però l'esquema permet triar altres tipus de materials (a la l'esquema es mostra amb una icona representant el node "choice").

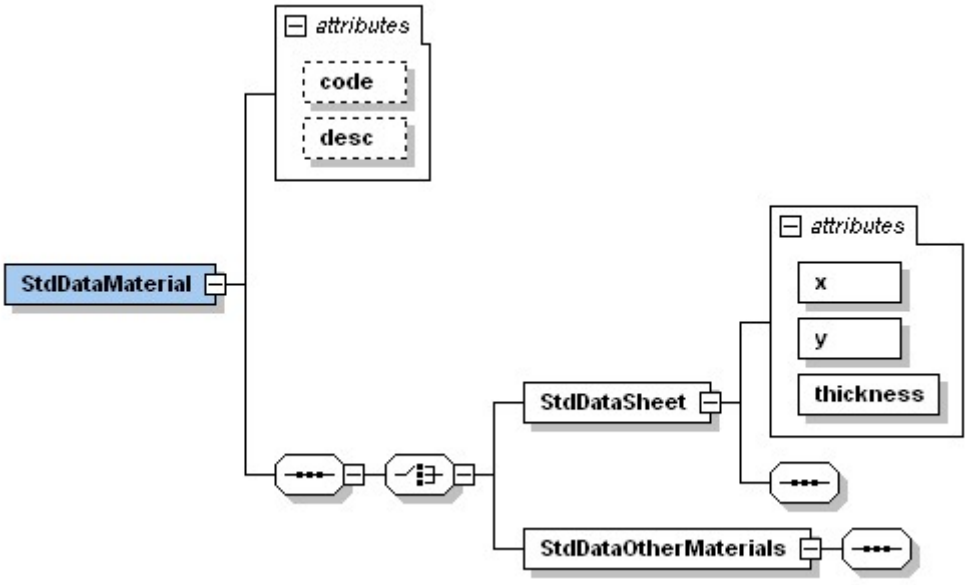


Figura 40. Esquema de tipus de material

- StdDataNesting (Figura 41): aquesta definició de l'esquema permet incloure una seqüència de 0 a 1000 peces (les que formen el nesting sobre el material, elements "StdDataPartNesting") . Cada peça requereix els atributs bàsics de dimensions que ocupa sobre la planxa ("x,y"), nombre de

vegades que apareix la peça sobre la planxa (atribut “num”) i el codi de la peça.

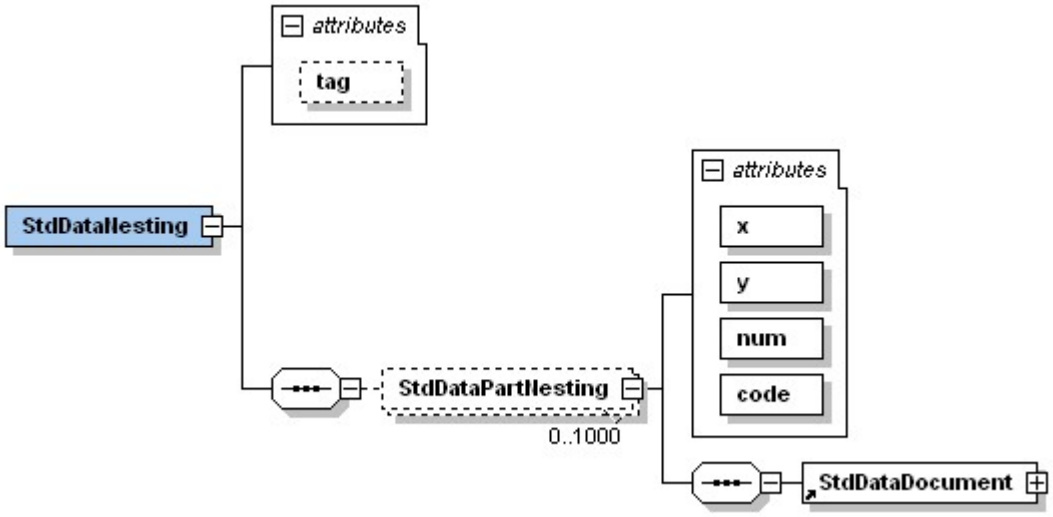


Figura 41. Peces del nesting

- StdDataEstimation (Figura 42): Opcional, són estimacions de fabricació per al programa. Inclou valors d’utilització de planxa, temps de treball, etc.

Figura 42. Estimacions de fabricació per al programa

- StdDataPunchingTools (Figura 43): en aquest cas es mostra la representació gràfica del fragment de definició que no s’ha mostrat a la Figura 37. Així es veu com, en funció de l’ utillatge escollit per a la màquina, cal introduir uns atributs diferents. Un punxó circular, per exemple, només està parametritzat pel seu diàmetre, mentre que un de rectangular depèn dels atributs “x” i “y”.

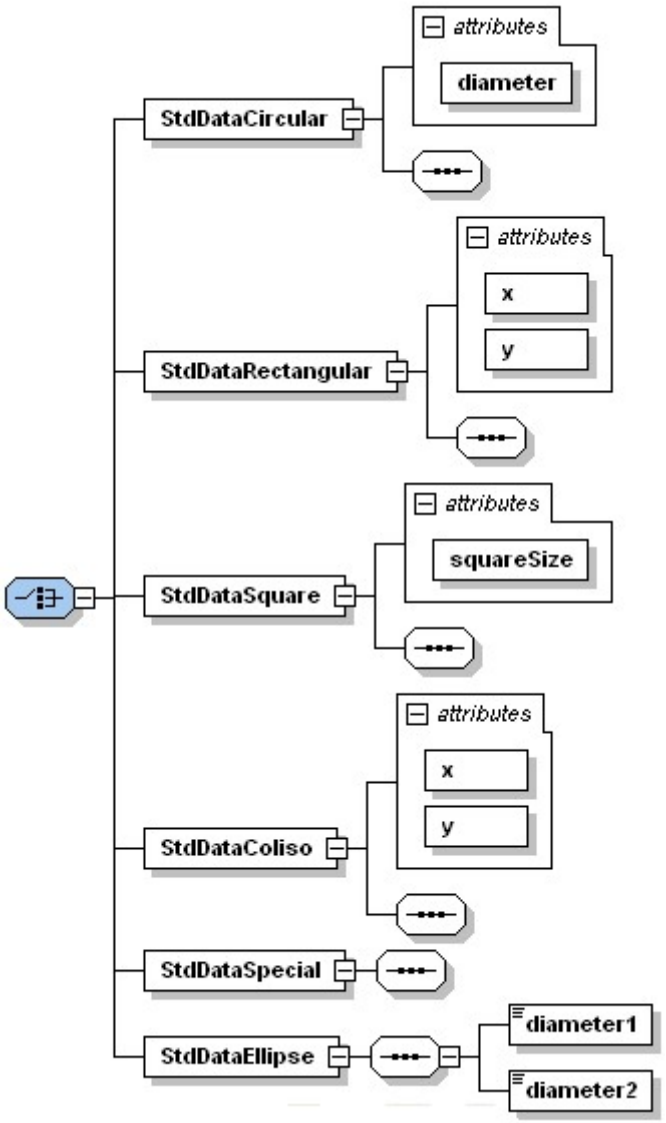


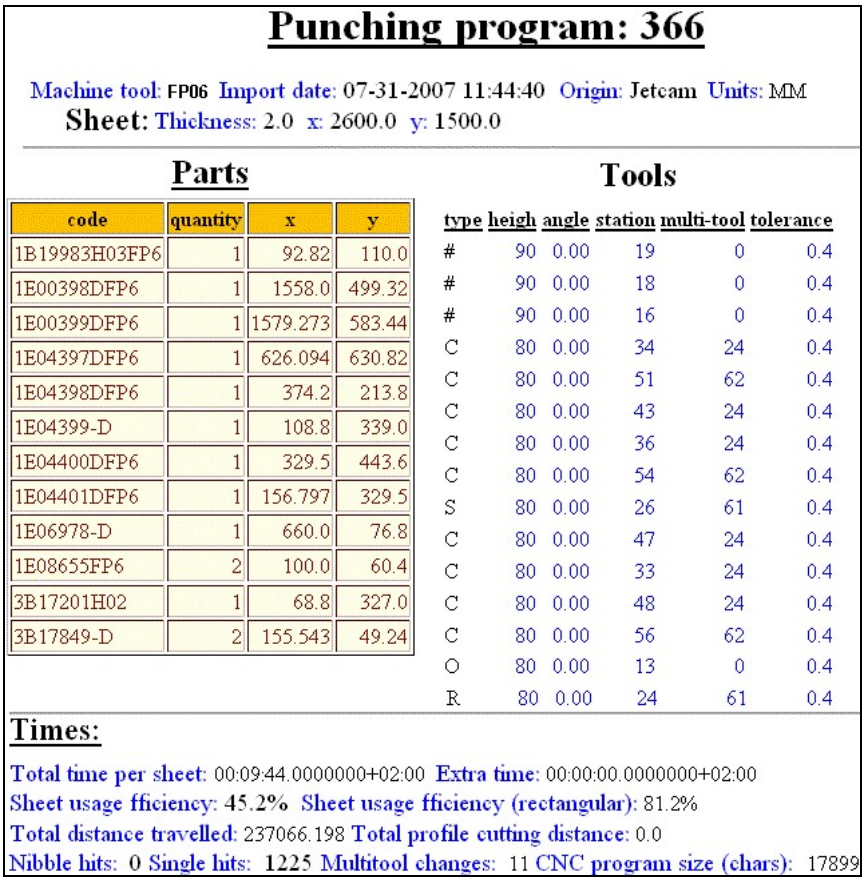
Figura 43. Esquema dels atributs pels utillatges de punxonadores

Després de veure els elements principals de la definició de l'esquema pels documents que contenen nestings en format XML, només queda mostrar un exemple de document vàlid per a l'esquema anterior, i aquest s'observa a la Figura 44. Es tracta d'un programa importat de JET-CAM®, on es fabriquen 3 peces sobre una planxa, amb la utilització de 3 utillatges.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="program.xslt" type="text/xsl"?>
<StdDataProgram >
  <StdDataControl importDate="2007-07-31T11:44:40.0204543+02:00" origen="Jetcam" />
  <StdDataGeneral code="366" units="MM">
    <StdDataMaterial>
      <StdDataSheet x="2600.0" y="1500.0" thickness="2.0" />
    </StdDataMaterial>
    <StdDataNesting>
      <StdDataPartNesting x="92.82" y="110.0" num="1" code="1B19983H03FP6">
      </StdDataPartNesting>
      <StdDataPartNesting x="626.094" y="630.82" num="3" code="1E04397DFP6">
      </StdDataPartNesting>
      <StdDataPartNesting x="374.2" y="213.8" num="1" code="1E04398DFP6">
      </StdDataPartNesting>
    </StdDataNesting>
  </StdDataGeneral>
  <StdDataEstimation sheetTime="00:09:44.0000000+02:00" extraTime="00:00:00.0000000+02:00"
toolChanges="11" cuttedDistance="0.0" travelledDistanced="237066.198" nibbleHits="0"
simpleHits="1225" sheetUsageEfficiency="45.2" sheetUsageEfficiencyRectangular="81.2"
cncProgramSize="17899" />
  <StdDataPunchingTools tolerance="0.4" station="34" angle="0.00" multi-tool="24" heigh="80">
    <StdDataCircular diameter="2.5" />
    <toolType>C</toolType>
  </StdDataPunchingTools>
  <StdDataPunchingTools tolerance="0.4" station="26" angle="0.00" multi-tool="61" heigh="80">
    <StdDataSquare costat="5.5" />
    <notes>SQR5.5</notes>
  </StdDataPunchingTools>
  <StdDataPunchingTools tolerance="0.4" station="9" angle="0.00" multi-tool="0" heigh="80">
    <StdDataRectangular x="10.1" y="4.1" />
    <toolType>R</toolType>
  </StdDataPunchingTools>
</StdDataProgram>
```

Figura 44. Exemple de document vàlid per a l'esquema de nestings

El darrer avantatge d'aquest tipus de representació estandarditzada és que és independent de la seva representació o interpretació. Aquest aspecte ja s'havia insinuat, però amb aquest exemple es veu com la integració de dades no només beneficia la part crítica que s'està adaptant (en aquest cas la incorporació de dades de nestings i estructures al sistema ERP). A la planta real, per exemple, s'han implementat fitxers de transformacions XSLT [79] per generar documents HTML dels nestings. D'aquesta manera, els programes generats o modificats es poden consultar per la intranet de l'empresa (i els operaris als terminals de planta) de forma immediata. Tornant a la Figura 44, l'etiqueta "xml-stylesheet" indica que aquest document es pot transformar amb les indicacions del fitxer "program.xslt". El resultat apareix a la Figura 45, on prèviament s'han afegit més peces i utillatges al programa.



Punching program: 366

Machine tool: FP06 Import date: 07-31-2007 11:44:40 Origin: Jetcam Units: MM
Sheet: Thickness: 2.0 x: 2600.0 y: 1500.0

Parts

code	quantity	x	y
1B19983H03FP6	1	92.82	110.0
1E00398DFP6	1	1558.0	499.32
1E00399DFP6	1	1579.273	583.44
1E04397DFP6	1	626.094	630.82
1E04398DFP6	1	374.2	213.8
1E04399-D	1	108.8	339.0
1E04400DFP6	1	329.5	443.6
1E04401DFP6	1	156.797	329.5
1E06978-D	1	660.0	76.8
1E08655FP6	2	100.0	60.4
3B17201H02	1	68.8	327.0
3B17849-D	2	155.543	49.24

Tools

type	heigh	angle	station	multi-tool	tolerance
#	90	0.00	19	0	0.4
#	90	0.00	18	0	0.4
#	90	0.00	16	0	0.4
C	80	0.00	34	24	0.4
C	80	0.00	51	62	0.4
C	80	0.00	43	24	0.4
C	80	0.00	36	24	0.4
C	80	0.00	54	62	0.4
S	80	0.00	26	61	0.4
C	80	0.00	47	24	0.4
C	80	0.00	33	24	0.4
C	80	0.00	48	24	0.4
C	80	0.00	56	62	0.4
O	80	0.00	13	0	0.4
R	80	0.00	24	61	0.4

Times:

Total time per sheet: 00:09:44.0000000+02:00 Extra time: 00:00:00.0000000+02:00
Sheet usage fficiency: 45.2% Sheet usage fficiency (rectangular): 81.2%
Total distance travelled: 237066.198 Total profile cutting distance: 0.0
Nibble hits: 0 Single hits: 1225 Multitool changes: 11 CNC program size (chars): 17899

Figura 45. Document HTML generat a partir de XML mitjançant XSTL[79]

5.4.4 Implementació del model d'agents simple

L'anàlisi, disseny i implementació del model d'informació estandarditzada (capítol 5.4.3) condiciona i facilita el procés d'implementació. Això es deu a que aquesta primera experiència real treballa amb pocs agents, i aquests tenen unes característiques molt simples. El fet de no haver disposat d'un middleware multiagent implantat a planta ha portat a la implementació de components ad-hoc que compleixen les propietats mínimes mostrades al capítol 2.5.1. Per fer-ho s'ha utilitzat un mecanisme de processos distribuïts (capítol 5.4.2) pels equips de planta. Com s'ha vist, aquests tenen un comportament actiu i es comuniquen utilitzant protocols estàndard (per exemple HTTP). És per això que es tracten com a agents, complint amb els requisits conceptuals del model de solució proposada. A més, com que els missatges que intercanvien són bàsicament contingut estandarditzat, es podran comunicar amb agents de plataformes estàndard o, simplement, es podrà substituir la part de comunicacions i *threading* del procés per la interfície amb la plataforma multiagent incorporada. Tenim, doncs, bona part de la implementació centrada en el tractament i manipulació de la informació, i en la distribució dels mòduls per ajustar-los al model teòric.

A la Figura 46 es mostra l'esquema de la part implementada on, d'acord amb el model teòric, s'han implementat els següents mòduls:

- NestingProgramSchema: aquest mòdul no és més que una llibreria de classes per manipular programes de nestings en format XML. Conté un conjunt de classes que representen els elements que composen un document de nesting en XML, d'acord amb l'esquema del capítol 5.4.3. Aquesta part de la llibreria s'ha generat mitjançant eines del VisualStudio 2005 que permeten generar codi font a partir dels esquemes XSD. Això representa diversos avantatges. El més clar de tots és que les classes generades són serialitzables, el programador només ha d'utilitzar la llibreria "System.XML.Serialization" del Framework per generar documents vàlids a partir dels objectes, o bé per generar una estructura d'objectes a partir d'un fitxer XML vàlid. Aquest tipus d'eines també existeixen en altres plataformes, per exemple JAXB [65] en Java. D'aquí que un bon disseny dels esquemes faciliti i condicioni la implementació d'aquesta part, si es canvia l'entorn de desenvolupament i/o plataforma es podran aprofitar, segurament, els esquemes per generar nou codi a partir d'ells.
- A més de les classes generades a partir dels esquemes, també s'inclouen altres classes amb funcions de manipulació i interfícies que implementaran posteriorment els Translators.
- FMSImporter: aquest mòdul implementa un Translator del model teòric. En aquest cas s'encarrega de la traducció dels nestings de JET-CAM® (fitxers amb extensió FMS, generats per macros) cap al format estàndard. El model d'execució distribuïda (capítol 5.4.2) permet que aquest Translator s'executi a l'equip on està instal·lat el mòdul de nestings i rebi peticions de traducció (de part dels LegacyWatchers) de fitxers allotjats al mateix equip. A més, incorpora llibreries que es poden utilitzar enllaçant directament amb els mètodes de conversió. Per exemple, el mètode "*clsXMLProgram fmsToProgram(String fileName)*" retorna l'objecte que representa el programa ("*clsXMLProgram*") a partir del nom del fitxer en format FMS. En aquest cas els esquemes han servit per detectar algunes incoherències entre les diferents bases de dades, problemes bàsics de gestió i organització. Per exemple, noms d'elements diferents al ERP i als mòduls de Nestings, codis inexistents en algun dels dos mòduls, etc. Com s'ha comentat anteriorment (5.4.3), no tots aquests tenen solució en aquest treball, sobretot els semàntics, però sí que s'ha ajudat a resoldre algun problema de redundància d'informació.
- ProgramImporter: el NestingImporter del model teòric s'ha implementat en aquest mòdul. Aquest és el que utilitza les llibreries d'accés al mòdul ERP, incorporant la informació generada pel FMSImporter. El fet de disposar de llibreries d'objectes d'accés al ERP ha facilitat la seva implementació. El mòdul pot rebre peticions de conversió del FMSImporter, i ha d'establir connexions amb la base de dades del ERP. En aquest cas ha de tractar problemes específics com, per exemple, la gestió de connexions amb llicència (es manté la connexió oberta si s'acumulen peticions en un interval curt de temps, es tanca si no hi ha peticions, etc.).

- FileSystemWatcher: en una implementació tan bàsica com aquesta, n'hi ha prou amb la detecció de canvis a un sistema de fitxers concret per implementar un dels LegacyWatchers del model. I així s'ha implementat el model d'agent, un procés que observa canvis al sistema d'arxius. En qualsevol moment pot rebre peticions d'"observar" canvis a altres sistemes d'arxius, o aturar la observació d'un d'ells. A diferència del model teòric, i un altre cop degut als condicionants tecnològics, en aquesta implementació no hi ha cap procés de diàleg o negociació per trobar els Translators corresponents. El fitxer de configuració (5.4.2) del FileSystemWatcher conté les URL dels servidors de Translators. En temps d'execució recupera aquesta informació i crea objectes proxy per als Translator remots. Cada vegada que detecta alguna modificació als fitxers dels sistemes llegats observats, passa la petició de traducció al proxy corresponent (cada Translator també pot informar del tipus de fitxer que converteix).

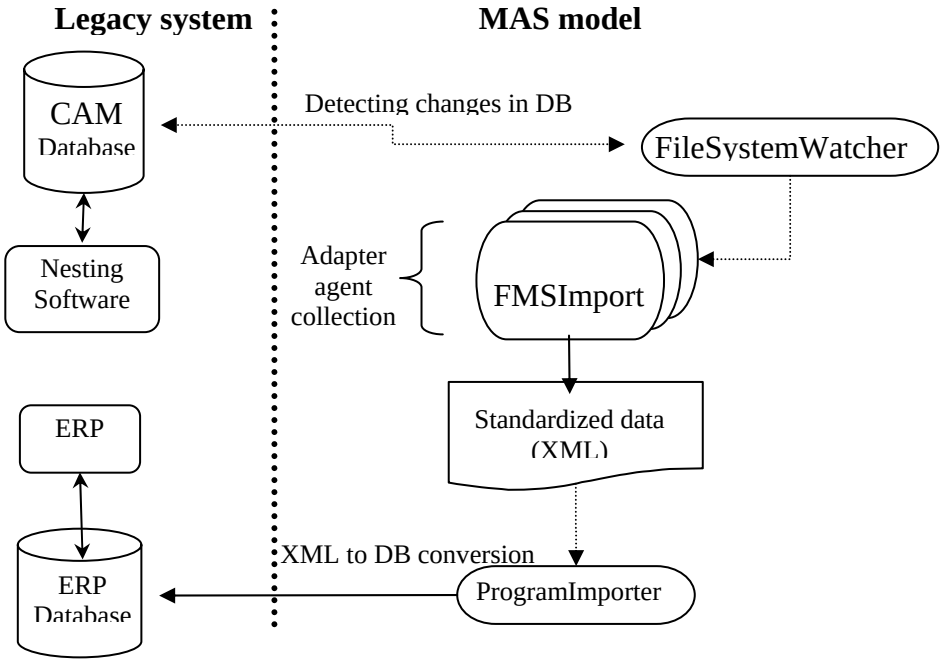


Figura 46. Esquema de la implementació per al mòdul de nesting

6. Conclusions

6.1 Motivació i objectius

A Catalunya, com a d'altres parts del món, hi ha un teixit gran de petites i mitjanes empreses que tenen unes plantes de producció altament tecnificades i que, consegüentment, s'enfronten al problema de l'actualització constant. Aquests canvis han de fer-se amb un cost assumible per a l'empresa sense que suposi un daltabaix en la producció que alteri la relació amb els clients, com per exemple, una baixada de la qualitat dels productes o un allargament de terminis de lliurament.

En aquest treball s'ha estudiat com fer servir els sistemes multiagent per solucionar el problema de l'actualització de les plantes de producció en petites i mitjanes empreses. La transformació del model de la planta en un model de sistema multiagent s'ha de fer tenint en compte que aquestes empreses no poden assumir els costos d'una aturada de la producció o de la construcció d'una planta alternativa i, per tant, s'ha de fer de forma progressiva i sense pertorbar (o amb la mínima pertorbació) el sistema productiu (es podria dir, "sense que aquests canvis afectin la relació amb el client").

L'objectiu d'aquest treball ha estat, doncs, d'investigar en com transformar l'organització d'una planta de producció d'una PME en un MAS que en permeti una actualització constant amb la mínima pertorbació.

Per a la realització d'aquest treball s'ha comptat amb la informació d'una planta de producció real en la que s'hi ha aplicat el que seria un primer pas en l'agentificació progressiva de la mateixa.

6.2 Treball realitzat

Per assolir l'objectiu principal d'aquest treball s'ha fet una revisió de l'estat de l'art per aquest problema multidisciplinari. Això ha implicat fer un recull de tecnologies, models teòrics i treballs en camps tan diversos com la informàtica industrial, la intel·ligència artificial, la robòtica o les xarxes. Tot això amb la intenció de poder aportar nocions d'aspectes concrets de planta (per exemple navegació de AGV, planificació de la producció o comunicació entre robots) al model de solució global que els hauria d'integrar. Una part de la revisió de l'estat de l'art aprofita l'experiència de l'autor d'aquest treball en la implementació d'un simulador de les comunicacions en colònies de robots modelats com a sistema multiagent. Tot i que no s'ha incorporat l'eina de simulació en aquesta implementació, i aquest treball no s'ha centrat en els aspectes de simulació, es podria plantejar la seva incorporació com a *middleware* d'alt nivell, o bé com a suport a la integració dels

subsistemes. De moment, fora d'aquest treball en una planta real, s'ha utilitzat en tres exemples teòrics de simulació [48][73][81]. Aquesta revisió de l'estat de l'art també ha permès fer una tria de tecnologies candidates a ser aplicades a planta i així proporcionar un entorn apropiat per al desenvolupament o aplicació dels possibles models de solució. El model de solució escollit per a la planta real consisteix en una transformació progressiva del sistema en funcionament (procés d'agentificació), i es proposen, com a metodologia simplificada, una sèrie de passos a seguir per aplicar aquesta solució. El pas següent ha estat obtenir el model de la planta real. De fet, el treball recull l'esquema de funcionament inicial i també el model funcional teòric de la planta "agentificada". Aquest pas d'anàlisi ha implicat parlar amb el personal de planta, cooperar amb els enginyers de l'empresa i seguir *in situ* el funcionament de la planta per tenir-ne la visió global. Finalment, s'ha fet una primera implantació parcial, molt reduïda, del model teòric. Aquesta primera implantació s'ha hagut d'ajustar als requeriments de planta com, per exemple, utilització de plataformes que utilitzen els seus enginyers, per a alguna part de la solució, temps disponible pel desenvolupament acotat, i necessitat de resultats ràpids. Treballant en un entorn real, aquest darrer requeriment, si s'assoleix, pot encoratjar l'ampliació del treball fet i impulsar línies de treball futures (un benefici "polític" de la confiança dels usuaris). Així doncs, s'ha hagut de sacrificar l'ús d'algunes tecnologies previstes al model de solució teòric per adaptar el treball a aquests requeriments.

Tot aquest procés de recerca permet treure algunes conclusions. Evidentment la primera d'elles és que el resultat pràctic de la primera prova d'implantació del model. Tot i que es tracta d'un treball molt preliminar, s'han aconseguit alguns resultats prou bons. El primer benefici, i més evident per als usuaris, ha estat l'estalvi de temps i agilització en el cicle de desenvolupament del producte a oficina tècnica. Només s'ha agentificat un dels mòduls software que generen els *nestings* (i els programes CN corresponents) per a 8 màquines eina de tallar i punxonar. Normalment el personal d'oficina tècnica crea o modifica uns 4 programes CN de mitjana al dia mitjançant aquest software. Després necessiten uns 15 minuts per introduir part de la informació d'aquests programes al sistema ERP. La implementació de la part del model teòric encarregada de la integració de *legacy systems* per aquesta part del sistema ha suposat l'estalvi d'unes 8 hores de treball humà al dia. A més d'aquests resultats més o menys quantificables, la consistència de la informació estandarditzada ha eliminat els temps perdut en detecció i correcció d'errors a la informació. Encara, més, la utilització de fitxers amb transformacions XSTL ha permès diversos tipus de visualització de la mateixa informació. El més pràctic ha estat de disposar a la intranet la versió en HTML dels documents estàndard tan bon punt es modifiquen. Aquests resultats han motivat que s'estiguin implementant dos "Translators" més: un pel software de *nesting* Bysoft i l'altre per l'eina CAD/CAM® Pro/Engineer®.

Més enllà dels primers beneficis empírics, es poden treure algunes conclusions sobre la metodologia i procés d'implantació proposats. L'experiència del treball realitzat mostra que el procés de creació del model teòric de planta és la part

fonamental. Encara que no s'aconseguís agentificar totalment la planta, aquest model és de gran utilitat ja que l'anàlisi de parts crítiques i la selecció de parts agentificables es fa en funció d'aquest model. A més, el sol fet de treballar en la construcció del model aporta informació de "reenginyeria" sobre els sistemes llegats. Aquí només s'ha mostrat un model de la planta de producció, en un primer nivell de refinament, però un modelat exhaustiu i molt més refinat pot requerir, a més de coneixements profunds de l'empresa, una gran quantitat de recursos de temps d'enginyers. És per això que caldria refinar més la metodologia simple proposada per a l'agentificació i especificar amb més detall com obtenir aquest model. Només aquest aspecte ja requereix ampliar la recerca en metodologies i eines de modelat i anàlisi del model de negoci (*workflows*, BPMN, simulació,...). També caldria detectar i analitzar els aspectes i processos comuns a aquest tipus d'empreses per veure si és possible trobar una generalització i estandardització de les diverses parts del model. Això facilitaria la tasca de modelat de la planta, aprofitant elements reutilitzables que modelin processos estàndard de negoci i utilitzant eines d'anàlisi automatitzades. Per exemple, la detecció de punts crítics al model, en aquest treball és un pas encara molt "artesanal", aquí també cal una recerca per veure si es pot automatitzar aquest procés. Aquests aspectes, possibles ampliacions de la recerca, podrien reduir els costos d'implantació d'aquest model de solució. Amb un temps de formació reduït, els enginyers de planta podrien prendre decisions ràpides sobre quines parts del sistema s'han de revisar o replantejar, sobretot si s'aconsegueix aquesta reducció del temps de modelat i, encara més, s'automatitzen les tasques d'anàlisi i detecció de punts crítics sobre el model. A més de la reducció de costos, el fet de tenir eines de modelat i anàlisi per treballar amb el model de planta proporciona fiabilitat a la metodologia, sobretot si aquestes tenen una base formal. Aquesta fiabilitat es podria traduir en seguretat per a les petites i mitjanes empreses a l'hora d'adoptar aquest tipus de solució amb un cost contingut.

Com hem vist, la implantació del model s'ha limitat a unes parts molt concretes i reduïdes, de components software i, a més, ha estat en molts aspectes "ad-hoc", limitada pels requeriments però intentant ajustar-se al model. Però això no ha impedit que els primers resultats aportin confiança als usuaris. Aleshores, apareix necessàriament una nova pregunta: si aquest model de solució aporta els beneficis vistos a nivell de planta de producció, es pot estendre a l'empresa sencera? I, en cas de que l'empresa en formi part, es pot estendre a una organització? Per respondre aquestes preguntes també caldria aprofundir més en la recerca. No tant, potser, en el model de solució en sí, sinó en mecanismes i problemes que s'han deixat de banda al tractar amb un entorn acotat. Un d'ells ja s'ha esmentat en aquest treball, el problema semàntic. En el cas de la planta aquest problema no ha aparegut perquè el sistema de representació de la informació i base de dades global s'han implementat de nou. Així que els Translators s'encarreguen d'ajustar les seves conversions a les regles gramaticals dels esquemes XSD i no hi ha problemes d'interpretació dels conceptes que hi apareixen i les relacions entre ells. A nivell d'empresa es podria fer alguna cosa

similar, però no així en el cas d'una organització. És més, si s'ha d'intercanviar informació amb altres empreses o altres organitzacions el problema apareix. En aquest sentit hi ha molta recerca en curs que tracten amb la utilització d'ontologies. Des de metodologies per implementar-les, llenguatges d'ontologies, taxonomies (des de simples llistes de conceptes i relacions entre ells a la utilització de bases formals complexes per especificar regles semàntiques) i, fins i tot, algunes bases d'ontologies públiques. A més, a nivell d'organització apareixen altres aspectes a estudiar com, per exemple, la seguretat. Si els agents estan oberts al món, caldrà estudiar quines relacions poden establir amb agents externs desconeguts, quina informació és permès oferir i, sobretot, preveure la possible existència d'agents "nocius" o perillosos pel sistema. Tot això dins un món d'organització de negocis cada cop més orientat a arquitectures de serveis i d'organitzacions que col·laboren entre elles (alguns d'aquests models de col·laboració encara estan en fase incipient de recerca com, per exemple, els ecosistemes digitals de negocis, "Digital Business Ecosystem" [54]).

Resumint, s'ha proposat un model de solució (l'agentificació) basat en el model multiagent, i s'ha proposat una metodologia simple a seguir pel cas de les plantes de producció petites i mitjanes. Els primers resultats mostren que és un model vàlid i aplicable, però la primera implantació ha requerit coneixement específic de l'entorn, sacrificar alguns punts de la metodologia i certa "habilitat" per implementar processos que es comportin com les entitats del model. Això obre múltiples possibles línies de treball. A més de seguir amb la implantació a una planta real i trobar problemes nous com, per exemple, el treball en temps real de màquines i robots (plataformes amb suport *real-time* i per agents físics), caldria aprofundir en la metodologia. Aquesta s'hauria refinar i proporcionar eines i paradigmes que permetin automatitzar part de la implantació del model, així com la seva anàlisi, a més de reduir-ne els costos d'implantació. Pel que fa a la planta real, com a línia de treball a curt termini es fa necessari implantar una plataforma multiagent estàndard. En aquest sentit ja s'ha mostrat que JADE compleix tots els requisits. En ser *open source* es pot integrar amb els mòduls i aplicacions existents i disposa d'una versió per dispositius mòbils (JADE-LEAP). A més, aquesta versió permet executar JADE sobre Microsoft .NET Framework, que és precisament la tecnologia sobre la que s'executa la implantació parcial de la planta (a més del ERP i altres mòduls existents). Finalment, encara no hi ha gaires experiències en la utilització de WADE (la versió 1.0 és del 05/05/2008), seria interessant aprofundir en l'execució de processos del negoci sobre un model de workflows, tot dintre d'una plataforma multiagent.

6.3 Línies de continuació

Tal com s'ha comentat, seria convenient refinar la metodologia de disseny i implementació que es proposa en aquest treball. Per fer-ho es podrien revisar i aprofitar metodologies conegudes de disseny de MAS (per exemple, la proposada a Zeus, entre d'altres), i mirar d'integrar-les amb metodologies i eines

especialitzades en modelat de processos de negoci i industrial. Així es podria obtenir una metodologia (amb eines) més completa i especialitzada en la creació de models de plantes de producció.

Seria interessant haver incorporat dades de comportament no funcional dels agents físics en el model per validació. En el treball efectuat s'ha tingut en compte especialment el flux d'informació i el software afectat perquè la implantació que l'empresa necessitava afectava components purament software. Entrant al terreny dels elements físics de la planta s'haurien de contemplar característiques com, per exemple, temps de procés unitari i de preparació per les màquines, paràmetres de resposta del robots, navegació, etc.

Estudi de casos de transformació en agents físics de determinades màquines. Per exemple, la transformació d'un carretó de càrrega (o una transpaleta) conduït per un humà en un agent del sistema (possiblement transformant-lo en un AGV). Això hauria de permetre aprofundir en aspectes com la integració de l'agent software sobre l'element físic (amb un PDA connectat directament, per exemple), o bé estudiar el cas en que l'agent software controlador està separat de la màquina física que controla. També es podrien estudiar opcions de transformació com, per exemple, un conjunt d'agents coordinats en el control d'una màquina, o un agent encarregat de controlar un grup de màquines.

Un cop transformada una planta en un MAS, inicialment tindrem un conjunt d'agents reactius que forment un sistema controlat globalment. Tots els elements de la planta estaran integrats i es podran comunicar d'una forma estandarditzada, mantenint una arquitectura modular. És a dir la funcionalitat dels subsistemes seguirà sent la mateixa, però s'hauran agilitzat i facilitat les relacions entre totes aquestes entitats mitjançant la utilització de la plataforma MAS com a middleware. A partir d'aquest punt es podrien començar a estudiar altres formes d'organització, donant més autonomia als agents. Això pot implicar la inclusió d'estratègies de negociació, la implementació d'un sistema holònic basat en els agents de planta, etc. En aquest cas la recerca es centraria en extreure models matemàtics que permetessin garantir el funcionament de la planta en qualsevol condició.

Apèndix A

Llista d'acrònims

ACL:	Agent Communication Language
AGV:	Automated Guided Vehicle
AOP:	Agent Oriented Programming
BDI:	Belief Desire Intention
BOM:	Bill Of Materials
BPMN:	Business Process Modeling Notation
CAD:	Computer-Aided Design
CAM:	Computer-Aided Manufacturing
CIM:	Computer Integrated Manufacturing
CL:	Content Language
CN:	Control Numèric
CNC:	Computer Numerical Control
DAI:	Distributed Artificial Intelligence
DCOM:	Distributed Component Object Model
DPS:	Distributed Problem Solving
DTD:	Document Type Definition
ebXML:	electronic business using Extensible Markup Language
EDI:	Electronic Data Interchange
ERP:	Enterprise resource planning
FIPA:	Foundation for Intelligent Physical Agents
FMS:	Flexible manufacturing system
FSM:	Finite State Machine
HW:	Hardware
HTTP:	Hyper Text Transfer Protocol
IIOP:	Internet Inter-ORB Protocol
JADE:	Java Agent DEvelopment Framework
JAXB:	Java Architecture for XML Binding
JIT:	Just-In-Time
JVM:	Java Virtual Machine

KQML:	Knowledge Query and Manipulation Language
KSE:	Knowledge Sharing Effort
LEAP:	Lightweight Extensible Authentication Protocol
LGPL:	Lesser General Public License
MAS:	Multi-Agent System
MRS:	Multi-Robot System
MRTA:	Multi-Robot Task Allocation
NC:	Numerical Control
NGMS:	Next Generation Manufacturing Systems
PDA:	Personal Digital Assistant
OF:	Ordre de Fabricació
PLC:	Programmable Logic Controllers
RMI:	Java Remote Method Invocation
RMS:	Reconfigurable Manufacturing Systems
RPC:	Remote Procedure Call
SPA:	Sense-Plan-Act Architectures
SQL:	Structured Query Language
SRS:	Single Robot System
STEP:	Standard for the Exchange of Product model data
SW:	Software
TCP:	Transmission Control Protocol
XML:	Extensible Markup Language
XSD:	XML Schema Definition
XSTL:	Extensible Style Sheet Language Transformations
UBL:	Unified Business Language
UN/EDIFACT:	United Nations/Electronic Data Interchange For Administration, Commerce and Transport
URI:	Uniform Resource Locator
WADE:	Workflows and Agents Development Environment (WADE)

Referències

[1] *AgentBuilder User's Guide*. Version 1.4. Acronymics, Inc. June,16, 2004.

[2] Arkin, R. C. "Motor schema-based mobile robot navigation", in *International Journal of Robotics Research* 8(4), pp. 9-12. 1989.

[3] Arkin, R.C., "Reactive Robotic Systems", article in Handbook of Brain Theory and Neural Networks, ed. M. Arbib, MIT Press, pp. 793-796, 1995.

[4] Arkin, R.C. and Balch,T.R. "Aura: principles and practice in review", in *Journal of Experimental and Theoretical Artificial Intelligence*, in press, 1997.

[5] Arkin, R. C. "Behavior-Based Robotics". Cambridge, MA, MIT Press. 1998.

[6] Austin, J. L. *How to do things with words*. Harvard University Press, 1962.

[7] Balch, T., Arkin, R.C. "Communication in reactive multi-agent robotic systems", in *Autonomous Agents*, 1, pp. 1-25, 1994.

[8] Basart, J.M. (2000) [1998]. Programació lineal. Col·lecció Materials de la UAB, n. 58. ISBN 84-490-1443-3.

[9] Bellifemine, F. ; Caire, G., et al. "JADE Programmers Guide". TILab. June-2007. Available: http://jade.tilab.com/doc/programmersguide.pdf.

[10] Berna-Koes, M.; Nourbakhsh,I. , and Sycara,K. "Communication Efficiency in Multi-Agent Systems", in Proceedings of ICRA 2004, New Orleans, LA, May 2004.

[11] Bonaveau, Eric; Dorigo, Marco; Theraulaz, Guy. *Swarm Intelligence. From Natural to Artificial Systems*. Oxford University Press. 1999.

[12] Borshchev, A. and Filippov A. From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools. Proceedings of the 22nd International Conference, Oxford, England, UK., July 25-29, 2004.

[13] Brooks, Rodney A. "A robust layered control system for a mobile robot", in *IEEE Journal of Robotics and Automation*, RA-2(1), pp. 14–23, April 1986.

[14] Brooks, Rodney A., "Challenges for complete creature architectures ", in From Animals to Animats: International Conference on Simulation of Adaptive Behavior, Jean A. Meyer and Stewart Wilson, Eds. MIT Press, 1990.

[15] Caire, G. "JADE Tutorial: JADE programming for Beginners". CSELTS.p.A. y TILab S.p.A. December 2003. Available: http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf.

[16] Caire, G. and Pieri, F. "LEAP USER GUIDE". Tilab January 2006. Revised on august 2008. Available: http://jade.tilab.com/doc/tutorials/LEAPUserGuide.pdf.

[17] Caliusco, M. L., Galli, M. R., Chiotti, O. "Information Integration Problem in B2B Relationships - A State-of-the-Art". In: Argentine Symposium of Software Engineering (32 JAIIO), 2003, Santa Fe (Argentina).

[18] Chase, R.B. and Aquilano, N.J., “Production and Operations Management”, Irwin, 5th edition, 1989.

[19] Collis, J. C. and Ndumu, D. T. “*The Role Modelling Guide”*. Informe. Applied Research and Technology, BT Labs. 1999.

[20] Darley, V. , “Emergent Phenomena and Complexity”, in R. Brooks & P. Maes, eds, Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, The MIT Press, 1994.

[21] Dartigues, C., Ghodous, P., Gruninger, M., Pallez, D., and Sriram, R. “CAD/CAPP Integration using Feature Ontology”. Concurrent Engineering: Research and Applications, Volume 15, Number 2, June 2007.

[22] De Vin, L. J.; Oscarsson, J.; Ng,A.; Jägstam, M and Karlsson,T., Manufacturing Simulation: Good Practice, Pitfalls, and Advanced Applications, published at 21st International Manufacturing Conference, pp. 156-163, Limerick, Ireland. 2004.

[23] De Vin, L.J.; Ng,A.H.C.; Oscarsson, J. and Andler,S.F., Information Fusion for Simulation Based Decision Support in Manufacturing, FAIM 2005 Special Issue of Robotics and Computer Integrated Manufacture, Vol 22, pp. 429-436, 2006.

[24] Dudek, Gregory; Jenkin, Michael; Milios, Evangelos and Wilkes, David. "A taxonomy for multi-agent robotics", in *Autonomous Robots*, vol. 3, pp. 375-397, 1996.

[25] Farinelli, A.; Iocchi, L. and D. Nardi, *Multirobot systems: A classification focused on coordination*. IEEE Trans. Syst., Man, Cybern. B, vol. 34, no. 5, pp. 2015-2028. 2004.

[26] FIPA Standard Specifications. Revised august 2008. Available: http://www.fipa.org/repository/standardspecs.html.

[27] FIPA Abstract Architecture Specification. Revised august 2008. Available: http://www.fipa.org/specs/fipa00001/SC00001L.html.

[28] Foundation for Intelligent Physical Agents (FIPA). Available: http://www.fipa.org/.

[29] Fowler, J. STEP for Data management, Exchange and Sharing, Great Britain: Technology Appraisals, 1995.

[30] García, A. and Cenjor, A."Sistema Heterárquico de Control Basado en Agentes para Sistemas de Fabricación: la Nueva Metodología PROHA". Revista Iberoamericana de Automática e Informática Industrial (RIAI). Enero 2007, núm. 1, pp. 83-94.

[31] Gat, E. (1992), "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots", in *Proceedings AAAI-92*, Vol. 4 No.4, pp.809-15.

[32] Gat, E. “On three-layer architectures”, in *Artificial Intelligence and Mobile Robots*. MIT/AAAI Press, 1997.

[33] Giret Boggino, Adriana. *Anemona: una metodología multi-agente para sistema holónicos de fabricación.* Universidad Politécnica de Valencia. PhD Thesis. Supervised by V. Botti. 2005.

[34] Giret Boggino, Adriana and Botti Navarro,Vicente J. “Aplicaciones Industriales de los Sistemas Multiagente”, *Agentes Software y Sistemas Multi-Agente. Conceptos, Arquitecturas y Aplicaciones.* pp. 186-203 . 2005.

[35] Goodrich, Michael. *Potential Fields Tutorial*. Class Notes. Available: http://www.ee.byu.edu/ugrad/srprojects/robotsoccer/papers/goodrich_potential_fields.pdf.

[36] Harrington J. *Computer integrated manufacturing*. New York: Industrial Press. 1973.

[37] Huhns, Michael N. and Stephens, Larry M. “Multiagent Systems and Societies of Agents” in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Weiss, G. Cambridge, MA: MIT Press, 2000, pp. 79-118.

[38] JADE, Java Agent DEvelopment Framework. Revised august 2008. Available: http://jade.tilab.com/.

[39] Jennings, Nicholas R.; Sycara Katia; Wooldridge, Michael. “A Roadmap of Agent Research and Development”, in *Autonomous Agents and Multi-Agent Systems*, 1, 1998, pp. 275–306.

[40] Jones, C.; Shell, D.; Mataric, M. J.; and Gerkey, B. 2004. “Principled approaches to the design of multi-robot systems, in *Proc. of the Workshop on Networked Robotics*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004).

[41] Klavins, Eric. "Communication Complexity of Multi-Robot Systems", in *Fifth International Workshop on the Algorithmic Foundations of Robotics*, Nice, France, Dec. 2002

[42] Kube, C. Ronald and Hong Zhang. Collective robotic intelligence. In Second International Conference on Simulation of Adaptive Behaviour , 1992, pp. 460–468.

[43] Labrou, Yannis. "Standardizing Agent Communication", in *Multi-Agent Systems & Application*, Advanced Course on Artificial Intelligence (ACAI-01) proceedings, Vladimr Marik and Olga Stepankova (editors), Springer-Verlag, 2001.

[44] Leszczyna,R. "Evaluation of agent platforms", European Commission, Joint Research Centre, Institute for the Protection and Security of the Citizen, Ispra, Italy, Tech. Rep., June 2004

[45] Lyons D. M., and Arbib, M. A. "A Formal Model of Computation for Sensory-Based Robotics", in *IEEE Trans. Rob. Aut.*, vol. 5, no. 3 (June 1989), pp. 280-293.

[46] MacLennan, B. “Synthetic Ethology: An Approach to the Study of Communication”. In *Articial Life II, SFI Studies in the Sciences of Complexity*, vol. XI, ed. Farmer et al, Addison-Wesley, 1991.

[47] Mataric, Maja J., *Interaction and Intelligent Behavior*, MIT AI Lab Tech Report # 1495, Aug 1994.

[48] Mauri, A; Trullàs, J and Ribas, Ll: "Un algoritmo de decisión para robots autónomos cooperantes para el desminado". In Workshop de Agentes Físicos 2006: pp. 119-126. April 2006.

[49] Mayfield, James; Labrou, Yannis and Finin, Tim. “Desiderata for agent communication languages” in *Proceedings of the 1995 AAAI Spring Symposium Information Gathering in Distributed Environments*, March1995.

[50] Mehrabi, M. G., Ulsoy, A. G., & Koren, Y. (2000). Reconfigurable manufacturing systems: key to future manufacturing. *Journal of Intelligent Manufacturing*, 11(3), 403-419.

[51] Microsoft Developer Network. “Información general de .NET Framework Remoting”. Revised august 2008. Available: http://msdn.microsoft.com/es-es/library/kwdt6w2k(VS.80).aspx.

[52] Microsoft Developer Network. "Overview of the .NET Framework". Revised august 2008. Available: http://msdn.microsoft.com/en-us/library/a4t23ktk.aspx.

[53] Muljadi, H.; Takeda, H. and Ando K. “Development of a Wiki-based Feature Library for a Process Planning System”, in Proceedings of the

7th International Conference on Enformatika Systems Sciences and Engineering (2005).

[54] Nachira, F.; Nicolai, A.; Dini,P.; Louarn,M. L. and Leon L. R.(editors). "Digital Business Ecosystems". European Commission, 2007. Revised august 2008 Available: http://www.digital-ecosystems.org/book/de-book2007.html.

[55] *NGMS-IMS Project. Interim Report* (online). Submitted April 2000. Revised august 2008. Available: http://www.cam-i.org/.

[56] *NGMS*-IMS Project. Phase III Final report (online). December 2005. Revised august 2008. Available: http://www.cam-i.org/.

[57] Nilsson, Nils J. *Shakey the Robot* (online). Technical note no. 323. April 1994. Revised august 2008. Available: http://www.ai.sri.com/shakey/.

[58] Nikraz, M; Caire, G and Bahri, PA . “A methodology for the analysis and design of multi-agent systems using JADE”. International Journal of Computer Systems Science and Engineering. 2006.

[59] Nwana, H. S., Ndumu, D. T., Lee, L. C., and Collis, J. C. “ZEUS: A Toolkit for Building Distributed Multi-Agent Systems”, in Applied Artificial Intelligence Journal, vol. 1, no. 13, pp. 129-185, 1999.

[60] OASIS (Organization for the Advancement of Structured Information Standards). "ebXML Business Process Specification Schema Technical Specification v2.0.4". OASIS Standard, 21 December 2006. Revised august 2008. Available: http://docs.oasis-open.org/ebxml-bp/2.0.4/OS/spec/ebxmlbp-v2.0.4-Spec-os-en-html/ebxmlbp-v2.0.4-Spec-os-en.htm.

[61] OASIS (Organization for the Advancement of Structured Information Standards). "Universal Business Language v2.0. Standard". 12 December 2006. Revised august 2008. Available: http://docs.oasis-open.org/ubl/os-UBL-2.0/UBL-2.0.html.

[62] Oliva Torras. Components metàl·lics. Revised august 2008. Available: http://www.olivatorras.com/archivoshtml/index.php?sel_lang=cat.

[63] OMG. Business Process Modeling Notation (BPMN) Specification 1.1.. February 2008. Revised august 2008. Available: http://www.bpmn.org/.

[64] Ontology bean generator [Online]. Revised august 2008. Available: http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator.

[65] Ort, E. and Mehta, Bhakti. "Java Architecture for XML Binding (JAXB)", in Sun Developer Network. March, 2003 (online). Revised august 2008. Available: http://java.sun.com/developer/technicalArticles/WebServices/jaxb/.

[66] Pechoucek,M.; Rehak, M. and Marik, V., "Expectations and Deployment of Agent Technology in Manufacturing and Defense: Case

Studies," Proc. 4th Int'l Conf. Autonomous Agents and Multi-Agent Systems—AAMAS 2005 Industry Track, ACM Press, 2005.

[67] Protege ontology editor [Online]. Revised august 2008. Available: http://protege.stanford.edu.

[68] Ross, Robert J. *MARC - Applying Multi-Agent Systems to Service Robot Control*. MSc Thesis. University College Dublin. 2003.

[69] Searle, John. *Speech Acts*. Cambridge Univesity Press, Cambridge, England, 1969.

[70] Shao, G., Lee, Y. T., and McLean C., “Manufacturing Data Validation Through Simulation”, in *Proceedings of the 2003 European Simulation and Modeling Conference*, pp 56-60, Naples, Italy, October 27-29, 2003.

[71] Sheth, A.P. "Changing Focus on interoperability in information systems: from system, syntax, structure to semantics". In Interoperating Geographic Information Systems, Eds Goodchild, M.., Egenhofer, M., Fegeas, R. & Kottman, C. , Klumer. 1998.

[72] Shoham, Y. “Agent-oriented programming”. *Artificial Intelligence*, 60(1), 1993, pp. 51-92.

[73] Trullàs, J. and Ribas, Ll. “Simulación de las comunicaciones en el sistema de robots hormiga ANTSYS”, in *VI Workshop en Agentes Físicos*, WAF2005, pp.75-82, Sept. 2005.

[74] Trullàs, J. and Ribas, Ll. "Experiences in the ‘Agentification’ of a Manufacturing Plant", in 2nd International IEEE Conference on Digital Ecosystems and Technologies (DEST2008). Phitsanulok / Thailand , 27-29 february 2008.

[75] Weitzenfeld, A., Arkin, R.C., Cervantes-Perez, F., Olivares, R., & Corbacho, F., A Neural Schema Architecture for Autonomous Robots, Proc. 1998 Int. *Symposium on Robotics and Automation*, Dec. 12-14, Saltillo, Coahuila, Mexico, 1998.

[76] Wooldridge, M. J. “Intelligent Agents” in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Weiss, G. Cambridge, MA: MIT Press, 2000, pp. 27-79.

[77] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Fourth Edition). Recommendation 16 August 2006, edited in place 29 September 2006 (online). Revised august 2008. Available: http://www.w3.org/TR/2006/REC-xml-20060816/.

[78] World Wide Web Consortium. “XML Schemas”. November 30, 2006 (online). Revised august 2008. Available: http://xml.coverpages.org/schemas.html.

[79] World Wide Web Consortium. "XSL Transformations (XSLT) Version 1.0". November 16, 1999 (online). Revised august 2008. Available: http://www.w3.org/TR/xslt.html.

[80] Workflows and Agents Development Environment (WADE). Revised august 2008. Available: http://jade.tilab.com/wade/.

[81] Yahia-Cheikh,N; Trullàs, J and Ribas, Ll: "Limpieza de superficies desconocidas mediante grupos de robots cooperantes". In Workshop de Agentes Físicos 2006: 135-142. April 2006.