UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Matheus Kraemer Bastos do Canto

A holonic manufacturing architecture for Line-less Mobile Assembly Systems operations planning and control

Florianópolis

2022

Matheus Kraemer Bastos do Canto

A holonic manufacturing architecture for Line-less Mobile Assembly Systems operations planning and control

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Engenharia de Automação e Sistemas.
Orientador: Professor Ricardo José Rabelo

Florianópolis

2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

do Canto, Matheus Kraemer Bastos do Canto
   A holonic manufacturing architecture for Line-less
Mobile Assembly Systems operations planning and control /
Matheus Kraemer Bastos do Canto do Canto ; orientador,
Ricardo José Rabelo , 2022.
   144 p.

   Tese (doutorado) - Universidade Federal de Santa
Catarina, Centro Tecnológico, Programa de Pós-Graduação em
Engenharia de Automação e Sistemas, Florianópolis, 2022.

   Inclui referências.

   1. Engenharia de Automação e Sistemas. 2. Line-Less
Mobile Assembly Systems. 3. Holonic Control Architecture.
4. Holonic Manufacturing Systems. 5. Ontology. I. ,
Ricardo José Rabelo. II. Universidade Federal de Santa
Catarina. Programa de Pós-Graduação em Engenharia de
Automação e Sistemas. III. Título.

Matheus Kraemer Bastos do Canto

A holonic manufacturing architecture for Line-less Mobile Assembly Systems operations planning and control

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof.(a) Ricardo José Rabelo, Dr.(a)
UNIVERSIDADE FEDERAL DE SANTA CATARINA

Prof.(a) Marcelo Ricardo Stemmer, Dr.(a)
UNIVERSIDADE FEDERAL DE SANTA CATARINA

Prof.(a) Jomi Fred Hübner, Dr.(a)
UNIVERSIDADE FEDERAL DE SANTA CATARINA

Prof.(a) João Carlos Espíndola Ferreira, Dr.(a)
UNIVERSIDADE FEDERAL DE SANTA CATARINA

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Engenharia de Automação e Sistemas.

ICP-Edu
Documento assinado digitalmente
Werner Kraus Junior
Data: 18/05/2022 09:13:43-0300
CPF: 531.085.239-53
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de Pós-Graduação

ICP-Edu
Documento assinado digitalmente
Ricardo Jose Rabelo
Data: 18/05/2022 11:05:42-0300
CPF: 481.802.119-91
Verifique as assinaturas em https://v.ufsc.br

Prof.(a) Ricardo José Rabelo, Dr.(a)
Orientador

Florianópolis, 2022.

Dedico esta dissertação de mestrado para minha mãe Míriam e meu pai Jefferson.

Acknowledgements

Sou imensamente agradecido ao Armin Buckhorst e ao Professor Ricardo José Rabelo aos quais devo todo o suporte técnico requisitado durante o desenvolvimento deste trabalho. Especialmente agradeço ao Armin Buckhorst por promover a oportunidade de uma colaboração entre a UFSC e o WZL da RWTH de Aachen. Estendo os especiais agradecimentos ao Professor Ricardo José Rabelo por acreditar nessa iniciativa e promover suporte, mesmo que de longe, durante os tempos difíceis em que vivemos.

Agradeço aos meus amigos Thiago Linhares Fernandes, Diogo de Carvalho Padilha e Maycon Luiz da Silva por permanecerem comigo e me incentivarem para superar os desafios acadêmicos e pessoais, mesmo através dos mais de 10.000 km que nos separaram quase que durante toda a construção desse trabalho. A meu amigo, que a Alemanha me deu, Presley, pelo seu companheirismo, sem o qual tudo teria sido extremamente mais difícil.

Não há medida que possa representar o quanto sou imensamente agradecido à minha namorada Aparna Joshi pelo seu essencial suporte emocional e técnico na variedade de desafios que o fato de morar noutro país nos obriga a enfrentar. Sou grato pelo acaso que nos uniu.

Agradeço à UFSC, ao PPGEAS e ao ensino público de educação brasileiro, no qual há muito tempo faço parte. O ensino médio e superior em instituições públicas de ensino me fez acreditar que a educação é base da qual necessitamos para resolver os mais intrincados dos problemas, sejam eles de natureza técnica ou social (principalmente).

Mesmo não conhecendo pessoalmente ainda a maioria dos colegas de mestrado, devido a característica especial deste trabalho e a pandemia, agradeço imensamente pelo companheirismo e pelo esforço despendido no exercício das tarefas e trabalhos em grupo, em especial à minha amiga Patrícia Mônica Campos Mayer Vicente. A todos os colegas e amigos que o contexto da UFSC me deu, meus maiores agradecimentos. Agradeço também ao departamento de Metrologia de Produção e Gestão da Qualidade do WZL, pelo acolhimento e suporte financeiro durante a minha estadia em Aachen. Obrigado aos meus amigos que WZL me deu: Mohamed, Gabriel, Javad, Richard, Danilo, Joaquin, Karen, Jan, Cristina, Karen e Jasmin pelas várias horas em discussões técnicas, small talk, tempo perdido, e almoços no Mensa.

À toda minha família, em especial meu pai Jefferson e minha mãe Míriam. Obrigado pela vida, ensinamentos, educação e por acreditarem que sempre há uma saída e se tentarmos e perseguirmos nossos sonhos. Consigo ver cada um de vocês dois na minha pessoa, e a cada dia, fico ainda mais grato pelo acaso ter me escolhido filho de vocês.

"The concept of hierarchic order occupies a central place in this book,
and lest the reader should think that I am riding a private hobby horse,
let me reassure him that this concept has a long and respectable ancestry.
So much so, that defenders of orthodoxy are inclined to dismiss it as "old hat",
and often in the same breath to deny its validity. Yet I hope to show as we go along
that this old hat, handled with some affection, can produce lively rabbits".
(Koestler, 1967)

Resumo

O Line-Less Mobile Assembly Systems (LMAS) é um paradigma de fabricação que visa maximizar a resposta às tendências do mercado através de configurações adaptáveis de fábrica utilizando recursos de montagem móvel. Tais sistemas podem ser caracterizados como holonic manufacturing systems (HMS), cujas chamadas holonic control architecture (HCA) são recentemente retratadas como abordagens habilitadoras da Indústria 4.0 devido a suas relações de entidades temporárias (hierárquicas e/ou heterárquicas). Embora as estruturas de referência HCA como PROSA ou ADACOR/ADACOR^2 tenham sido muito discutidas na literatura, nenhuma delas pode ser aplicada diretamente ao contexto LMAS. Assim, esta dissertação visa responder à pergunta "Como uma arquitetura de produção e sistema de controle LMAS precisa ser projetada?" apresentando os modelos de projeto de arquitetura desenvolvidos de acordo com as etapas da metodologia para desenvolvimento de sistemas holônicos multi-agentes ANEMONA. A fase de análise da ANEMONA resulta em uma especificação do caso de uso, requisitos, objetivos do sistema, simplificações e suposições. A fase de projeto resulta nos modelos de organização, interação e agentes, seguido de uma breve análise de sua cobertura comportamental. O resultado da fase de implementação é um protótipo (realizado com o Robot Operation System) que implementa os modelos ANEMONA e uma ontologia LMAS, que reutiliza elementos de ontologias de referência do domínio de manufatura. A fim de testar o protótipo, um algoritmo para geração de dados para teste baseado na complexidade dos produtos e na flexibilidade do chão de fábrica é apresentado. A validação qualitativa dos modelos HCA é baseada em como o HCA proposto atende a critérios específicos para avaliar sistemas HCA. A validação é complementada por uma análise quantitativa considerando o comportamento dos modelos implementados durante a execução normal e a execução interrompida (e.g. equipamento defeituoso) em um ambiente simulado. A validação da execução normal concentra-se no desvio de tempo entre as agendas planejadas e executadas, o que provou ser em média irrelevante dentro do caso simulado considerando a ordem de magnitude das operações típicas demandadas. Posteriormente, durante a execução do caso interrompido, o sistema é testado sob a simulação de uma falha, onde duas estratégias são aplicadas, LOCAL_FIX e REORGANIZATION, e seu resultado é comparado para decidir qual é a opção apropriada quando o objetivo é reduzir o tempo total de execução. Finalmente, é apresentada uma análise sobre a cobertura desta dissertação culminando em diretrizes que podem ser vistas como uma resposta possível (entre muitas outras) para a questão de pesquisa apresentada. Além disso, são apresentados pontos fortes e fracos dos modelos desenvolvidos, e possíveis melhorias e idéias para futuras contribuições para a implementação de sistemas de controle holônico para LMAS.

Keywords: Line-Less Mobile Assembly Systems. Holonic Control Architecture. Holonic Manufacturing Systems. Ontology.

Resumo expandido

Introdução

Nos últimos anos, a diversidade de preferências dos clientes levou ao aumento das variedades de produtos, resultando na redução do tamanho dos lotes das empresas e na urgência de lidar com a personalização em massa (OLIVEIRA, 2019). As altas exigências dinâmicas dos clientes, o avanço dos conceitos de fabricação e a disponibilidade cada vez maior de técnicas sofisticadas de Tecnologia da Informação (TI) influenciaram diretamente os antigos paradigmas de fabricação, tais como Sistemas de Manufatura Dedicada (DMS) e Sistemas de Montagem Dedicada (DAS), para evoluir da rigidez para paradigmas de fabricação mais flexíveis. Exemplos destas abordagens são os Sistemas de Manufatura Flexível (FMS) e os Sistemas de Manufatura Reconfiguráveis (RMS). Sob a perspectiva da montagem, abordagens equivalentes são respectivamente os Sistemas de Montagem Flexíveis (FAS) e os Sistemas de Montagem Reconfiguráveis (RAS) (BUCKHORST *et al.*, 2019).

Embora os novos paradigmas de fabricação e montagem tenham proposto flexibilidade crescente, seus conceitos dependem de estações de trabalho estacionárias e níveis de célula modular, restringindo a capacidade de abordar trabalhos individuais e rotas de peças (BUCKHORST *et al.*, 2019). A disposição das etapas do produto durante a realização da montagem, a diversidade das capacidades de montagem, a generalização desafiadora em meio a diferentes operações do produto e a individualização do produto dificultam a implementação do RAS. Para lidar com tais desafios, a implementação de uma nova abordagem chamada Sistemas de Montagem móveis sem linha (LMAS) é pesquisada e desenvolvida na cadeira de Metrologia de Produção e Qualidade do Laboratório de Máquinas-Ferramenta da Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen (Werkzeugmaschinenlabor der RWTH Aachen).

A evolução da indústria, chamada Indústria 4.0, é caracterizada por um alto nível de recursos conectados ao longo de todo o ciclo de vida do produto (YANG; GU, 2021). A riqueza da disponibilidade de dados em todo o fluxo de produtos abre novas oportunidades de negócios e possibilidades para melhorar a cadeia de processo agregando valor ao produto final (TAY *et al.*, 2018). Entre outras suposições, a Indústria 4.0 considera que as empresas devem planejar e executar seu processo com adaptabilidade às flutuações externas/internas em mente (AHELEROFF *et al.*, 2020). Tais flutuações assumem vários formatos, tais como mudanças na demanda de mercado, pedidos apressados e relações com concorrentes de mercado (OLIVEIRA, 2019). Neste contexto, especialmente para as empresas de manufatura, torna-se vital considerar o uso de sistemas de informação conectados a arquiteturas de controle interoperáveis, flexíveis e reativas (DERIGENT *et al.*, 2020).

Antigos sistemas de controle de fabricação centralizados não foram concebidos para se adaptarem às mudanças e exigências intensas do mercado atual, como personalização em massa, ciclo de vida curto do produto e preços de produtos concorrentes (BARBOSA *et al.*, 2015). Novas abordagens mais recentes, seguindo uma organização heterárquica, foram propostas. Em uma organização heterárquica, o problema de controle é resolvido através da cooperação entre os componentes do sistema, em vez de um plano rígido fornecido por uma entidade superior do sistema. A vantagem mais significativa é que os sistemas heterárquicos se adaptam prontamente às interrupções do sistema (por exemplo, pedidos urgentes, defeitos de produtos, falhas de recursos).

Neste contexto, uma solução que une as vantagens das abordagens hierárquica e heterárquica toma forma nas arquiteturas de controle holônico (HCA) (BARBOSA *et al.*, 2015; LEITÃO; RESTIVO, 2006; BRUSSEL *et al.*, 1998; JIMENEZ *et al.*, 2017) que se baseiam no conceito do sistema de Manufatura Holônica (HMS).

O LMAS é um paradigma de montagem projetado para grandes produtos (HÜETTEMANN *et al.*, 2019) e consiste em três princípios-chave (BUCKHORST *et al.*, 2020): (a) abordagem de piso limpo; (b) mobilidade de todas as entidades; (c) tarefas totalmente flexíveis: (HÜETTEMANN *et al.*, 2019). Um problema de pesquisa ainda em aberto lida com o controle das operações LMAS, exigindo uma arquitetura de sistema capaz de lidar com a reconfigurabilidade topológica, planejamento e controle da LMAS. Esforços na modelagem dos conceitos, estrutura e implementação de LMAS foram aplicados (HÜETTEMANN *et al.*, 2019; BUCKHORST *et al.*, 2020; BUCKHORST *et al.*, 2019). Assim, uma arquitetura de planejamento e controle é obrigatória para gerenciar a atribuição ideal de montagem/fluxo de peças através de recursos, executar etapas de produção e responder a interrupções nos sistemas, tais como novos pedidos de clientes ou falhas de equipamentos. Esta dissertação é baseada neste problema aberto específico e propõe uma arquitetura HMS adequada para o LMAS. Assim pretende-se responder a pergunta científica: **Como a arquitetura de um sistema de produção e controle LMAS precisa ser projetada?**

Objetivos

O objetivo geral é projetar, implementar e avaliar uma arquitetura holônica para o planejamento e controle das operações do sistema de montagem móvel sem linha (LMAS).

Objetivos específicos

- definição da metodologia de projeto do HMS;
- projeto e implementação de um modelo de dados baseado em uma ontologia de nível superior para apoiar a aplicação de arquitetura;
- projeto da arquitetura HMS composta por:
  - modelo de organização;
  - modelo de agente;
  - modelo de tarefa/objetivo;
  - modelo de interação;
  - modelo de ambiente.
- implementação da arquitetura do HMS em forma de protótipo de sofware utilizando framework de desenvolvimento para robótica;
- integração do protótipo de arquitetura com módulos anteriores do LMAS (por exemplo: algoritmo de otimização);
- avaliação da lógica do software do protótipo e das características holônicas propostas em um ambiente simulado;
- análise quantitativa e qualitativa dos ensaios simulados, a fim de apoiar a resposta a ser dada à questão da pesquisa;

Metodologia

Este projeto de pesquisa foi realizado nas instalações da WZL localizada em Aachen, Alemanha. A pesquisa será desenvolvida através da colaboração com o PPGEAS da UFSC e o MAA da WZL. O trabalho geral é dividido em cinco fases principais:

- revisão da literatura;
- projeto de fabricação holônica;
- implementação da fabricação holônica;
- avaliação da arquitetura holônica proposta;
- análise da arquitetura holônica proposta.

Em uma visão geral, embora metodologias distintas sejam aplicadas em cada fase, todo o projeto é inspirado na metodologia pesquisa-ação, onde o objetivo é a resolução de um problema particular (SILVA; MENEZES, 2005).

De acordo com a metodologia de pesquisa-ação: o problema é primeiramente identificado, depois é seguido por um estudo que deriva um conjunto de possíveis soluções para esse problema. As soluções são analisadas e uma é levada à ação. Os dados que saem da experiência são utilizados para avaliar as conseqüências sobre o sistema. De acordo com as conseqüências, os resultados são interpretados à luz do sucesso da ação. O ciclo se repete sistematicamente até que o problema seja resolvido.

Resultados e discussões

O resultado da fase de implementação é um protótipo (realizado com o Robot Operation System) que implementa os modelos ANEMONA e uma ontologia LMAS, que reutiliza elementos de ontologias de referência do domínio de manufatura. A fim de testar o protótipo, um algoritmo para geração de dados para teste baseado na complexidade dos produtos e na flexibilidade do chão de fábrica é apresentado. A validação qualitativa dos modelos HCA é baseada em como o HCA proposto atende a critérios específicos para avaliar sistemas HCA. A validação é complementada por uma análise quantitativa considerando o comportamento dos modelos implementados durante a execução normal e a execução interrompida (e.g. equipamento defeituoso) em um ambiente simulado. A validação da execução normal concentra-se no desvio de tempo entre as agendas planejadas e executadas, o que provou ser em média irrelevante dentro do caso simulado considerando a ordem de magnitude das operações típicas demandadas. Posteriormente, durante a execução do caso interrompido, o sistema é testado sob a simulação de uma falha, onde duas estratégias são aplicadas, LOCAL_FIX e REORGANIZATION, e seu resultado é comparado para decidir qual é a opção apropriada quando o objetivo é reduzir o tempo total de execução. Finalmente, é apresentada uma análise sobre a cobertura desta dissertação culminando em diretrizes que podem ser vistas como uma resposta possível (entre muitas outras) para a questão de pesquisa apresentada. Além disso, são apresentados pontos fortes e fracos dos modelos desenvolvidos, e possíveis melhorias e idéias para futuras contribuições para a implementação de sistemas de controle holônico para LMAS.

Considerações finais

Esta dissertação apresenta uma arquitetura de controle holônico desenvolvida para o LMAS respeitando suas exigências e características operacionais específicas. Primeiramente, uma revisão de bibliografia foi conduzida a fim de se identificar obras relacionadas a metodologia de desenvolvimento holônicas para sistemas multi-agentes, arquiteturas de controle holônicas, ontologias e requisitos do LMAS. Com base nos resultados da revisão bibliográfica e dos requisitos do LMAS, apresenta-se os modelos de organização, interação, tarefa/objetivo, ambiente e agente, acompanhados de um protótipo de software seguindo as definições nos modelos. Em seguida um modelo de dados baseado em ontologia e um protótipo de sistema seguindo as recomendações da metodologia da ANEMONA. O protótipo foi avaliado durante a execução normal e interrompida em ambiente simulado.

O HCA permite a autonomia das entidades LMAS e permite a reconfiguração espaço-temporal da fábrica e da estação. A abordagem ontológica para implementar o modelo de dados permitiu a identificação das interações e atributos necessários para soluções baseadas em agentes e promove a interoperabilidade semântica. A avaliação do desempenho do protótipo durante a execução normal indica aderência à reconfigurabilidade desejada e estabilidade adicional. Como pretendido, os holons da camada de execução seguem seus cronogramas locais, com atrasos de execução insignificantes, mesmo tendo miopia ao plano de fabricação global. No caso da execução interrompida, pôde-se estabelecer um trade-off entre as duas estratégias utilizadas para resolução de falhas (REORGANIZATION e LOCAL_FIX), comparando o aumento no tempo de execução entre as duas estratégias em casos de falhas simuladas. Como contribuições futuras, os resultados apresentados neste trabalho sugerem a necessidade de um tratamento mais poderoso das interrupções de operações que ocorrem na prática industrial (por exemplo, alocação de operações com base em alternativas em suas seqüências de operação); a comparação da execução da arquitetura com uma abordagem de ambiente simulado DES, que é uma ferramenta que pode ser útil quando se trata de simulação de incertezas de execução e comportamentos imprevistos; a tomada de decisão entre as estratégias disponíveis de tratamento das interrupções é um problema a ser investigado; em outras palavras, como decidir seguir uma estratégia específica em vez de outras baseadas em objetivos de fabricação pré-definidos (por exemplo durante todo o processo, tempo de conclusão, tempo de transporte); e o comportamento do protótipo de arquitetura quando conectado ao equipamento real, o que imporia um aumento nos atrasos e robustez para lidar com as incertezas do processo de montagem real.

Palavras-chave: Sistema de montagem móvel sem linha. Arquitetura de Controle Holônico. Sistemas Holônicos de Manufatura. Ontologia.

Abstract

The Line-Less Mobile Assembly Systems (LMAS) is a manufacturing paradigm aiming to maximize responsiveness to market trends (product-individualization and ever-shortening product lifecycles) by adaptive factory configurations utilizing mobile assembly resources. Such responsive systems can be characterized as holonic manufacturing systems (HMS), whose so-called holonic control architectures (HCA) are recently portrayed as Industry 4.0-enabling approaches due to their mixed-hierarchical and -heterarchical temporary entity relationships. They are particularly suitable for distributed and flexible systems as the Line-Less Mobile Assembly or Matrix-Production, as they meet reconfigurability capabilities. Though HCA reference structures as PROSA or ADACOR/ADACOR^2 have been heavily discussed in the literature, neither can directly be applied to the LMAS context. Methodologies such as ANEMONA provide guidelines and best practices for the development of holonic multi-agent systems. Accordingly, this dissertation aims to answer the question "How does an LMAS production and control system architecture need to be designed?" presenting the architecture design models developed according to the steps of the ANEMONA methodology. The ANEMONA analysis phase results in a use case specification, requirements, system goals, simplifications, and assumptions. The design phase results in an LMAS architecture design consisting of the organization, interaction, and agent models followed by a brief analysis of its behavioral coverage. The implementation phase result is an LMAS ontology, which reuses elements from the widespread manufacturing domain ontologies MAnufacturing's Semantics Ontology (MASON) and Manufacturing Resource Capability Ontology (MaRCO) enriched with essential holonic concepts. The architecture approach and ontology are implemented using the Robot Operating System (ROS) robotic framework. In order to create test data sets validation, an algorithm for test generation based on the complexity of products and the shopfloor flexibility is presented considering a maximum number of operations per work station and the maximum number of simultaneous stations. The validation phase presents a two-folded validation: qualitative and quantitative. The qualitative validation of the HCA models is based on how the proposed HCA attends specific criteria for evaluating HCA systems (e.g., modularity, integrability, diagnosability, fault tolerance, distributability, developer training requirements). The validation is complemented by a quantitative analysis considering the behavior of the implemented models during the normal execution and disrupted execution (e.g.; defective equipment) in a simulated environment (in the form of a software prototype). The normal execution validation focuses on the time drift between the planned and executed schedules, which has proved to be irrelevant within the simulated case considering the order of magnitude of the typical demanded operations. Subsequently, during the disrupted case execution, the system is tested under the simulation of a failure, where two strategies are applied, LOCAL_FIX and REORGANIZATION, and their outcome is compared to decide which one is the appropriate option when the goal is to reduce the overall execution time. Ultimately, it is presented an analysis about the coverage of this dissertation culminating into guidelines that can be seen as one possible answer (among many others) for the presented research question. Furthermore, strong and weak points of the developed models are presented, and possible improvements and ideas for future contributions towards the implementation of holonic control systems for LMAS.

Keywords: Line-Less Mobile Assembly Systems. Holonic Control Architecture. Holonic Manufacturing Systems. Ontology.

List of Figures

List of Tables

List of abbreviations and acronyms

HCA	Holonic Control Architecture
HMS	Holonic Manufacturing Systems
LMAS	Line-Less Mobile Assembly System
MAS	Multi-Agent Systems
FMS	Flexible Manufacturing System
RMS	Reconfigurable Manufacturing Systems
FAS	Flexible Assembly Systems
RAS	Reconfigurable Assembly Systems
IT	Information Technology
DMS	Dedicated Manufacturing Systems
UFSC	Federal University of Santa Catarina
RWTH	Rheinisch-Westfälische Technische Hochschule
ERP	Enterprise Resource Planning
MES	Manufacturing Execution System
IMS	Intelligent Manufacturing Systems
PPGEAS	Graduate Program in Automation and System Engineering
MAA	Metrology Assisted Assembly Group
WZL	Werkzeugmaschinenlabor der RWTH Aachen
CPS	Cyber-Physical Systems
CPPS	Cyber-Physical Production Systems
CS	Computer Science
ICT	Information and Communication Technologies
NSF	National Science Foundation

PCL	Programmable Computer Logic
EC	European Community
EFTA	European Free Trade Association
PROSA	Product Resource Order Staff Architecture
ADACOR	ADAptative holonic COntrol aRchitecture
HCBA	Holonic Component Based Architecture
PROSIS	Product, Resource, Order, Simulation for Isoarchy Structure
ORCA	Optimized and Reactive Control Architecture
ANEMONA	A Multi-agent Methodology for Holonic Manufacturing Systems
DACS	Designing Agent-based Control Systems
ROMAS	Regulated Open Multi-Agent Systems
ARTI	Activity, Resource, Type and Instance Architecture
CASE	Computer-Aided Software Engineering
DAI	Distributed Artificial intelligence
AGV	Automated Guided Vehicle
UC	Use Case
RQ	Requirement
RDF	Resource Description Framework
SPARQL	Protocol and RDF Query Language
OWL	Web Ontology Language
SKOS	Simple Knowledge Organization System
LDE	Local Decisional Entities
RDE	Resource Decisional entities
GDE	Global Decisional Entities
SoA	Service-oriented Architecture
DES	Discrete-event Simulation

SoHMS	Service-oriented Holonic Manufacturing Systems
AI	Artificial Intelligence
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
OCHRE	Object-Centered High-level Reference Ontology
SUMO	Suggested Upper Merged Ontology
BFO	Basic Format Ontology
FIPA	Foundation for Intelligent Physical Agents
H2CM	Holonic Hybrid Control Model
MaRCO	Manufacturing Resource Capability Ontology
MASON	MAnufacturing's Semantics ONtology
MSDL	Manufacturing Service Description Language
SA	Simplification & Assumption
FLERMO	FLExible Resource Manufacturing Ontolog
HOLLONTO	HOLonic assembLy manufacturing ONTOlog
Holontology	Holonic ontology
ROS	Robot Operating System
JADE	Java Agent DEvelopment Framework
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
SMH	Switch Mechanism Holon
OH	Order Holon
SH	Station Holon
AEH	Assembly Equipment Holon
PH	Part Holon
SAH	Sub-assembly Holon
AH	Assembly Holon

Contents

1 Introduction

In recent years, customer preferences diversity has led to increased product varieties, resulting in reduced companies' lot sizes and the urgency to cope with mass customization (OLIVEIRA, 2019). The high dynamic customer requirements, the advancement of manufacturing concepts, and the ever-increasing availability of sophisticated Information Technology (IT) techniques directly influenced the former manufacturing paradigms, such as Dedicated Manufacturing Systems (DMS) and Dedicated Assembly Systems (DAS), to evolve from rigidity to more flexible manufacturing paradigms. Examples of these approaches are the Flexible Manufacturing Systems (FMS) and Reconfigurable Manufacturing Systems (RMS). Under the assembly outlook, equivalent approaches are respectively Flexible Assembly Systems (FAS) and the Reconfigurable Assembly Systems (RAS) (BUCKHORST *et al.*, 2019). Although the new manufacturing and assembly paradigms have proposed increasing flexibility, their concepts rely on stationary workstations and modular cell levels, constraining the ability to address individual job and part routes (BUCKHORST *et al.*, 2019). The arrangement of product steps during assembly realization, diversity of machining assembly capabilities, challenging generalization amid different product operations, and product individualization hamper a RAS implementation. To handle such challenges, the implementation of a novel approach called Line-Less Mobile Assembly Systems (LMAS) has been researched and devised at the chair of Production Metrology and Quality of the Machine Tool Laboratory of the Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen (Werkzeugmaschinenlabor der RWTH Aachen).

1.1 General problem

The evolution of industry, so-called, Industry 4.0 is characterized by a high level of connected resources along all the product life cycle (YANG; GU, 2021). The richness of data availability across the product flow opens new business opportunities and possibilities to enhance the process chain adding value to the final product (TAY *et al.*, 2018). Among other assumptions, Industry 4.0 considers that companies should plan and execute their process with adaptability to external/internal fluctuations in mind (AHELEROFF *et al.*, 2020). Such fluctuations assume various formats such as market demand changing, rush orders, and market competitors relations (OLIVEIRA, 2019). In that context, especially for the manufacturing companies, it becomes vital to consider the use of information systems connected to reactive interoperable and flexible control system architectures (DERIGENT *et al.*, 2020).

The control system architectures can be understood as control systems responsible for defining the system's composition elements, operation behaviors, and control execution. In practice, in the case of a manufacturing system, the instantiation of a control system architecture takes form in a Manufacturing Execution System (MES) that connected to the Enterprise Resource Planning (ERP), governs the manufacturing operation in turn planning, control, and execution levels (JIMENEZ *et al.*, 2017).

Initially, a significant amount of solutions for such control system architecture have been devised to be centralized architectures. One great advantage of a centralized architecture is that it can be optimized globally. It can easily query the whole system state because the system's components are all known to the central controller and derive an optimal global plan based on the current state. With the centralized strategy, the optimal plan deployment is executed via a hierarchical approach where the components strictly obey the given plan. The most significant disadvantage of a centralized solution is the lack of adaptability and reactivity capabilities (BARBOSA *et al.*, 2015).

Former centralized manufacturing control systems have not been devised to adapt to the current high intense market-changing and requirements like mass customization, short product life-cycle, and competing product prices (BARBOSA *et al.*, 2015). Industry 4.0 holds the promise of increased flexibility (AHELEROFF *et al.*, 2020) and adaptability, which requires control architectures to deal with high dynamic scenarios. In general, (YANG; GU, 2021) lists five particular points that Industry 4.0 drags attention to:

- shortening the development period with the use of highly innovative means;
- using ultra-customization to end the traditional "one for all" and to promote uniqueness or is sometimes called "batch size one" in manufacturing;
- productions are integrated with higher flexibility;
- enabling faster decision-making procedures by decentralization as opposed to lengthy organizational hierarchy;
- promoting sustainability and resource efficiency in the context of the ecological aspect.

More recent new approaches following a heterarchical organization have been proposed. In a heterarchical organization, the control problem is solved via the cooperation between the system components instead of a strict plan supplied by a higher system entity. The most significant advantage is that the heterarchical systems adapt promptly to system disruptions (e.g., rush orders, product defects, resource faults). The main disadvantage is the lack of performance in providing an optimal global plan due to the system myopia (one entity has a limited capacity regarding the system perception) (WANG; HAGHIGHI, 2016).

In this context, a solution that couples the advantages from both hierarchical and heterarchical approaches takes form in the holonic control architectures (HCA) (BARBOSA *et al.*, 2015; LEITÃO; RESTIVO, 2006; BRUSSEL *et al.*, 1998; JIMENEZ *et al.*, 2017) which are based on the Holonic Manufacturing system (HMS) concept. The HMS offers a conceptualization to derive HCAs that benefit from the best of the heterarchical and hierarchical approaches. The HMS concept utilizes the idea of autonomous entities called holons (CHRISTENSEN, 1994). These entities are organized assuming the most appropriate behaviors, which may vary between heterarchical and hierarchical depending on the system state. The name given to this dynamic organization is holarchy (BRUSSEL *et al.*, 1998).

In this work, a control architecture for the LMAS paradigm is derived based on the HMS concept.

1.2 Specific problem

The LMAS is an assembly paradigm designed for large products (HÜETTEMANN *et al.*, 2019) and consists of three key principles (BUCKHORST *et al.*, 2020): (a) clean floor approach; (b) mobility of all entities; (c) fully flexible assignments: (HÜETTEMANN *et al.*, 2019). Applying these principles provides a dynamic response that allows reconfigurability and scalability to meet high altering market demands. These three principles are summarized in the figure 1.

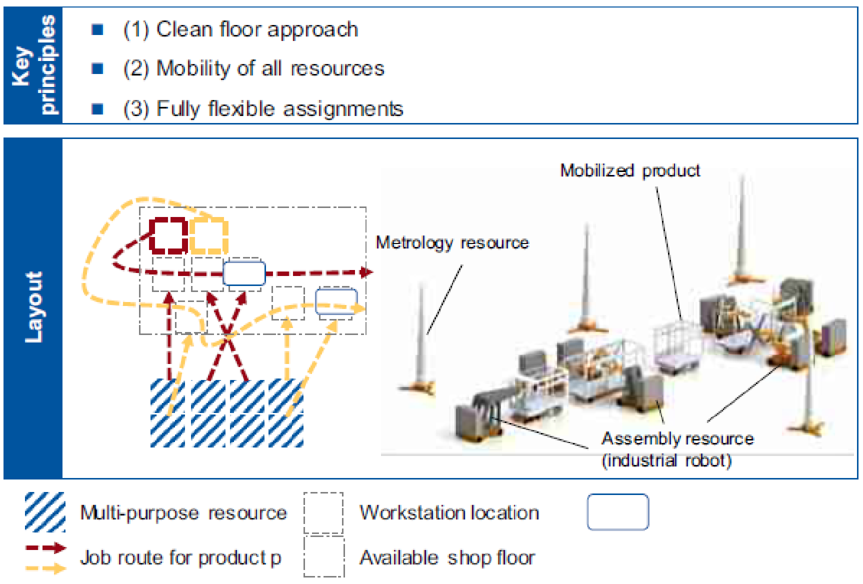


Figure 1 – LMAS three key principles overview, adopted from (HÜETTEMANN *et al.*, 2019)

Efforts in modeling the LMAS concepts, framework, and implementation have been applied (HÜETTEMANN *et al.*, 2019; BUCKHORST *et al.*, 2020; BUCKHORST *et al.*, 2019). An example of the connection between the HMS concepts and the LMAS is presented in (BUCKHORST *et al.*, 2021, in Press). In this work, the LMAS system architecture is composed of three layers: the planning layer, the control layer, and the execution layer. In each layer, holons are designed to take functionalities such as planning and scheduling (planning layers), control and monitoring (execution layer), and operation execution (execution layer). The figure 2 exhibits the LMAS architecture separated into layers and highlights its main components.

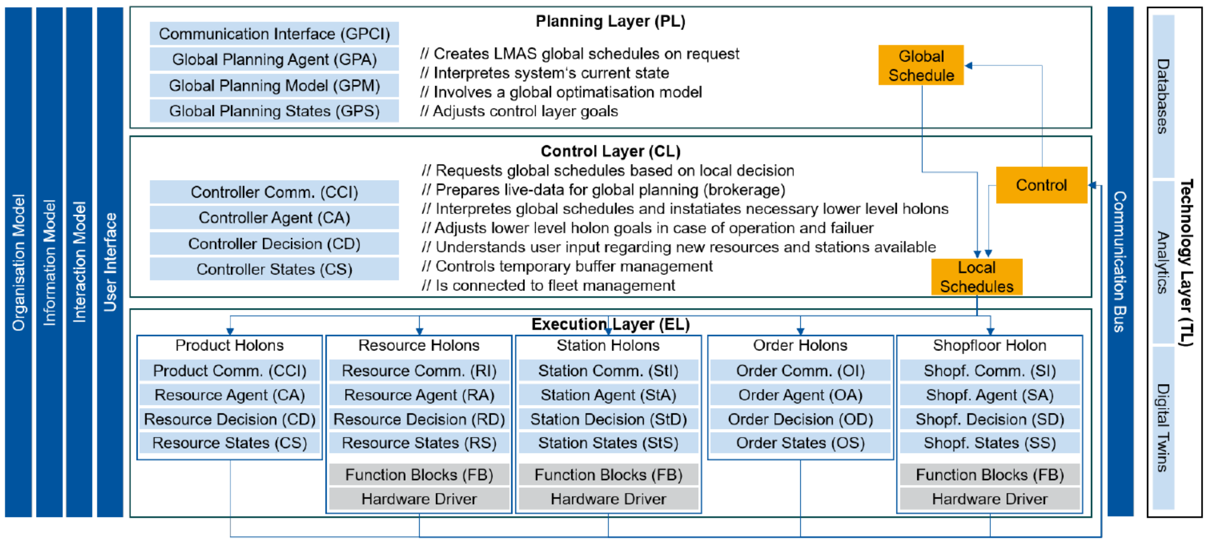


Figure 2 – LMAS architecture, adopted from (BUCKHORST *et al.*, 2021, in Press)

A remaining open research problem deals with LMAS operations control, requiring a system architecture capable of dealing with LMAS's topological reconfigurability, planning, and control. Accordingly, a planning and control architecture is mandatory to manage the optimal assembly/part flow assignment through resources, execute production steps, and respond to disruptions to the systems, such as new customer orders or equipment faults. This dissertation is based on this specific open problem and proposes a suitable HMS architecture for the LMAS.

The HMS proposal was built over the Intelligent Manufacturing Systems (IMS) requirements for 21st-century manufacturing systems. Hence it promises to offer a suitable control strategy adaptable to LMAS control and planning main requirements, given in the format of the three already presented principles.

The present dissertation, envisioning to evaluate the HMS concept applied for the LMAS paradigm, investigates, proposes, and tests an architecture consisting of interaction, agent, and organization models based on the holonic manufacturing system concept. The performance shall be evaluated via simulation. The current work can be viewed as a continuation of the research started on (BUCKHORST *et al.*, 2021, in Press). The main differences are related to:

- the interaction model of the assembly equipment holons. The station described in (BUCKHORST *et al.*, 2021, in Press) is considered an instantiation of assembly equipment, while on this current work, the author proposes to increment the interaction model in order to support the communication required to create dynamic stations composed of a variety of assembly equipment;

- the data model classes. The data model classes structure should be adapted to represent the shop floor granularity, the holons states, the necessary data representation to support the addition to the interaction model, the scheduler, the sub-assembly, and the order holons. The data model derived in this dissertation will be based on a higher level manufacturing domain ontology, as suggested by the HMS methodology used;
- the control layer interaction model. It is necessary to propose and increment modifications to the control layer to fulfill the holon's initialization, addition, deletion, customer order interpretation, simulation execution, and optimization layer integration (input/output data interpretation and distribution to operation holons). For example, the new broker holon's addition will create, monitor, and delete all the holon classes. The broker holon is essential on the initialization phase and acts as the principal holon on the control layer;
- integration between modules related to LMAS former initiatives. This work integrates the optimization, control, and operation layers and validates the results in a simulated environment. The simulation is carried with a simple timing structure, where wait operations in the prototype replace the assembly and transport operations;
- disruption orchestration: this work considers in the architecture design models the fact that the system should correctly respond to disruption such as equipment fault. This work proposes a simple treatment for fault detection and correction based on two simple strategies: complete reorganization and local fix (equipment replacement). The orchestration is carried out by adding a new holon, the switch mechanism holon.

1.3 Research question

How does an LMAS production and control system architecture need to be designed?

1.4 Justification

The increase in product customization, shortened product life-cycle, and high competition between companies to attend product lower prices are characteristics of Industry 4.0. Consequently, manufacturing companies should implement flexibility, scalability, and modularity in their production systems to cope with the Industry 4.0 requirements. The HMS concept envisions to support dynamic organization structures where a hierarchical and heterarchical approach is applied to adapt the production systems to incoming orders and system faults. The HCA has its basis on the HMS concepts, and it might offer a suitable architecture metamodel for the novel LMAS manufacturing paradigm.

Nevertheless, this research project aims to design and evaluate a suitable HCA applied for the novel LMAS manufacturing paradigm. This work will also contribute to HMS concerning evaluating the efficiency of using heterarchical and decentralized control architectures to respond to disruptions in the production execution phase (e.g., resource unavailable, rush orders, upcoming new production batches).

Besides the necessary and essential support of the Graduate Program in Automation and System Engineering (PPGEAS) of the Federal University of Santa Catarina (UFSC), the conduction of this research is supported by the Group Metrology Assisted Assembly (MAA) of WZL.

1.5 Objective

1.5.1 General objective

Design, implement and evaluate a holonic architecture for Line-Less Mobile Assembly System operations planning and control.

1.5.2 Specific objectives

Considering the LMAS requirements and the general objective presented, the following specific objectives are highlighted:

- HMS design methodology definition;
- design and implementation of a data model based in a higher-level ontology to support the architecture application;
- design of HMS architecture composed of:
    - organization model;
    - agent model;
    - task/goal model;
    - interaction model;
    - environment model.
- implementation of the HMS architecture utilizing a robotic software framework platform as a prototype;
- integration of the architecture prototype with former modules of LMAS (e.g.: optimization algorithm);

- evaluation of the prototype's software logic and proposed holonic features in a simulated environment;
- quantitative and qualitative analysis of the simulated trials in order to support the answer to be given to the research question;

1.6 Scientific contribution

This work's main contribution to the scientific community is related to a proposal of a holonic control architecture for the LMAS paradigm. In order to contribute to previous approaches for a suitable holonic control architecture for LMAS, this work:

- proposes architecture models developed following a methodology specific for HMS concepts;
- adds to the previous contribution the necessary interactions to create dynamic stations (interaction model);
- utilizes a data model based on a higher manufacturing ontology artifact;
- adds a simple disruption orchestration.

This work's progress and intermediate results were submitted in the format of a scientific paper in the 55th CIRP Conference on Manufacturing Systems 2022 (CMS2022), under the title "A Holonic Control System Approach for Line-Less Mobile Assembly Systems".

1.7 Organization

The remainder of this work is organized as follows: chapter two presents the applied methodology on research and work; chapter three presents a theoretical background on essential concepts related to this research work; chapter four presents the literature review; chapter five presents the control architecture design models; chapter six covers the implementation aspects of the HCA; chapter seven covers the test and results; and chapter eight overviews the developed HCA models and presets and answer to the research question.

2 Research and work methodology

2.1 Research methodology

This research project will be conducted in the physical facilities of the WZL located in Aachen, Germany. The research will be constructed by collaborating with the UFSC's PPGEAS and the WZL's MAA. The overall work is divided into five main phases.

In an overview, though distinct methodologies are applied in each phase, the whole project is inspired in the Action-Research methodology, where the objective is the resolution of a particular problem (SILVA; MENEZES, 2005).

The action-research methodology cycle is presented in the figure 3. According to the presented diagram, the problem is first identified, then it is followed by a study that derives a collection of possible solutions for that problem. The solutions are analyzed, and one is taken to action. The data that comes out of the experiment is used to evaluate the consequences on the system. According to the consequences, findings are interpreted in the light of how successful the action was. The cycle repeats systematically until the problem is solved.

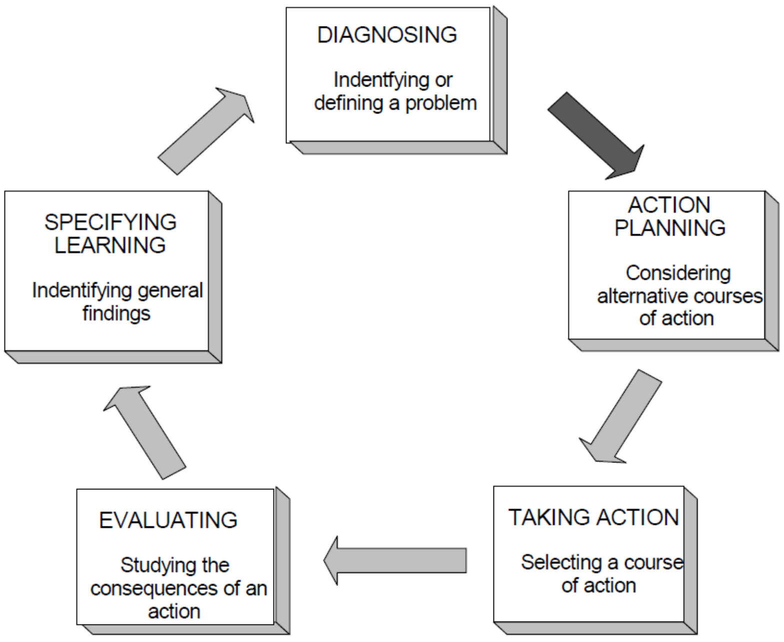

DIAGNOSING
Indentfying or defining a problem
ACTION PLANNING
Considering alternative courses of action
TAKING ACTION
Selecting a course of action
EVALUATING
Studying the consequences of an action
SPECIFYING LEARNING
Indentifying general findings

Figure 3 – Action Research: methodology cycle, adopted from (YASMEEN, 2008)

Subsequently, the general methodology is briefly described:

- objective: Action-Research. The research is realized with a specific action or a problem resolution in mind. All participants are cooperatively involved in solving this specific action or problem;
- nature: applied. The research aims to generate knowledge for practical applications oriented to solve a specific problem;
- research method: Quali-quantitative. The research aims to propose a holonic architecture that should be evaluated qualitatively and quantitatively. The generated design models and concepts are in the qualitative evaluation while the system prototype is on the quantitatively part;
- procedures: literature review, experiments driven by simulations, and use cases analysis;
- local: The research will be conducted in the WZL laboratory located at Aachen, Germany.

The research methods applied in each research phase are described below:

- **literature review**: the review will be held on public and private consultation bases over books, articles, dissertations, and master/bachelor dissertation. The candidate will conduct this review with the PPGEAS and WZL teams' support and guidance. It is intended to cover the last 20 years of research on the field. The literature review phase is related to the Diagnosing and Action Planning phases of each iteration in the action research methodology;
- **holonic manufacturing design**: design of the architecture models composing the holonic control architecture for LMAS. The design will be described through a representational suggested by the HMS methodology applied. The holonic manufacturing design phase is related to the Action Planning phase of each iteration in the action research methodology;
- **holonic manufacturing implementation**: on this phase, agile software methodology, each identified software module is implemented. This phase is related to the Taking Action phase of each iteration in the action research methodology;
- **evaluation of the proposed holonic architecture**: the evaluation of the proposed HCA will be conducted with two objectives: first, evaluate the architecture design models, and second the prototype performance accordingly to the identified use cases. The design models are evaluated concerning their performance, reliability, availability, modifiability, and functionality. The proposed holonic architecture prototype will be evaluated by applying the use cases utilizing a simulated environment. As an input for the use cases, genuine industrial pumps data will simulate the incoming customer orders. The evaluation of the

proposed holonic architecture phase is related to the Evaluating phase of each iteration in the action research methodology;

- **analysis of the proposed holonic architecture**: in this phase, a quantitative and qualitative analysis of the proposed simulated prototype outputs will be conducted. The result of that phase is the argumentation for the research question answer. The analysis of the proposed holonic architecture phase is related to the Specifying Learning phase of each iteration in the action research methodology.

2.2 Work methodology

During the progress of this work, a methodology inspired by agile methodologies (e.g., SCRUM, Kanban, Scrumban) will be taken to deal with the programming and research tasks.

The student will conduct regular bi-weekly meetings with the local supervisor to keep up with the last findings and new resolutions. This approach is beneficial in multiple ways, as the actual research project has requirements that can affect parallel developments and vice-versa.

During the bi-weekly meetings, the current tasks will be reviewed and new ones created depending upon the actual need or last discussions. Tools for task management (e.g., Trello) and version control systems (e.g., git, Gitlab) are used and, when necessary, provided by the WZL infrastructure.

To keep up with the UFSC's Professor advisor, Professor Rabelo, monthly meetings will be held between all the parts: the candidate student, the local advisor, and Professor Rabelo. In these meetings, the objective is to report the project's current status, share knowledge, and get insights to better conduct the dissertation outcome.

Besides the monthly meetings, emails will be sent between the candidate student and the Professor to answer specific dissertation questions.

3 Theoretical background

3.1 Industry 4.0

The Industry 4.0 definition must be clarified in this work as the HMS itself is an enabling technology for its requirements. Industry 4.0 concept is a consequence of a new industrial revolution. However, three revolutions shaped how society consumes, communicates, and conducts business along with modern history. Industrial revolutions are associated with technological advancement but bound to societal requirements (e.g., sustainability). The paradigm shift acts like a trigger that instigates new business models and affects industrial sites and innumerable society layers (OLIVEIRA, 2019).

The first industrial revolution (1760 - 1840) was characterized by the steam machine and the train rails with which the manual production force was escalated to an increased level of productivity. The second industrial revolution is marked mainly by introducing electrical energy, making it possible to develop the production line concept and mass production. The third industrial revolution, 1960, is associated with electronics, informatics, and internet development and usage. These achievements are the base for the always-increasing sharing of knowledge via the network and the digital transformation of information (TAY *et al.*, 2018).

The entire chain of industrial development served as a basis to establish the mechanisms for the emerging Industry 4.0 revolution. Industry 4.0 is usually associated with manufacturing, though it is not restricted to that context. Industry 4.0 addresses a wide range of topics involving innumerable technologies convergence, process automation, efficient use of energy, sustainability, and increasing of autonomous systems and Cyber-Physical Systems (CPS) (SCHWAB, 2016).

In (ZEZULKA *et al.*, 2016) the authors developed a 3D model for Industry 4.0 that represents the interconnection between technical and business features envisioned by this model. The 3D model is observed in figure 4: the vertical axis represents the viewpoints about the benefits of Industry 4.0 from the business level to the asset level; the left horizontal axis represents the stream of the product life-cycle, where there are opportunities to collect valuable data to improve not only the product but the whole process; the right-horizontal axis describes the functional position of components, aiming to identify where the function is allocated (e.g., product, enterprise, stations).

For an enterprise to benefit from what Industry 4.0 promises, it is necessary to adapt to a set of technical requirements. These requirements are (HERMANN *et al.*, 2016; LAMNABHI-LAGARRIGUE *et al.*, 2017):

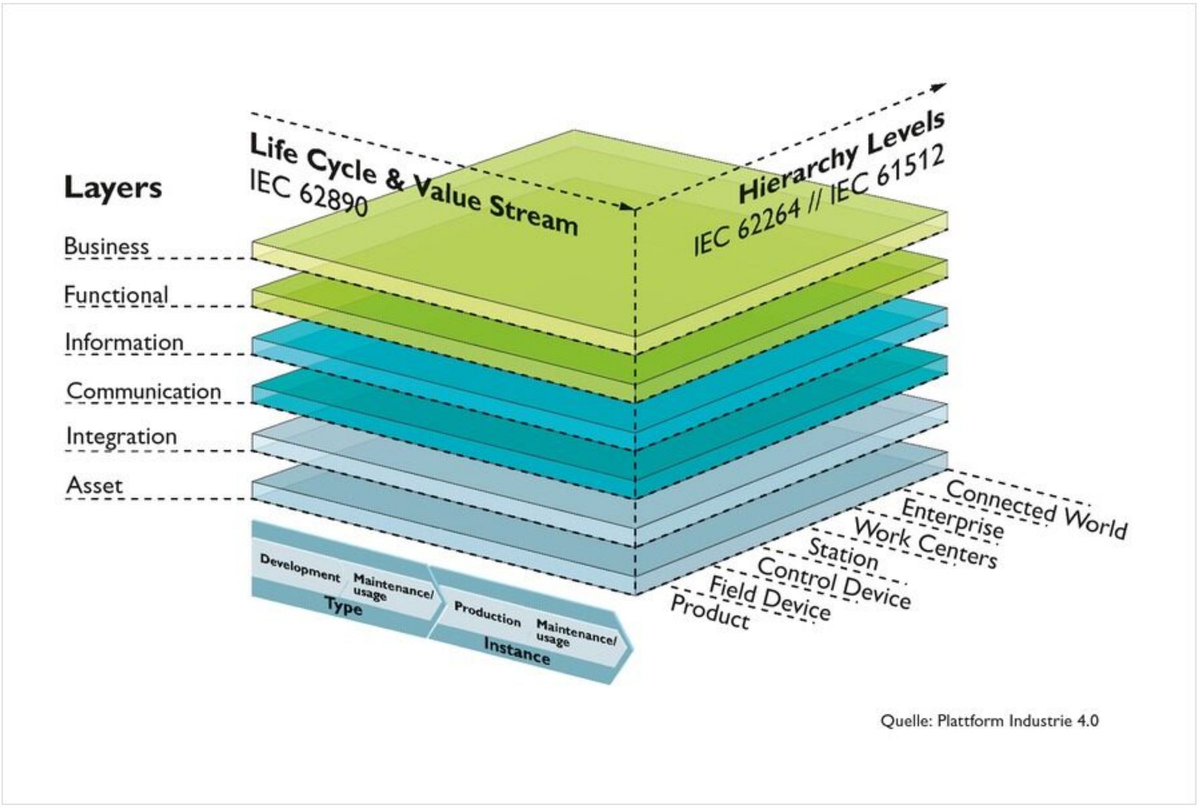


Figure 4 – RAMI: the reference architectural model for Industry 4.0, adopted from (ZEZULKA *et al.*, 2016)

1. distributed and decentralized control;
2. decision autonomy;
3. CPS collaboration;
4. virtualization of organizational entities of any type and level;
5. adaptability and ability of company and systems to be plug & play;
6. interoperability between distributed systems and heterogeneous technologies;
7. cybersecurity;
8. emergent behavior and dynamic self-organization;
9. resilient system supervision and operation;
10. real-time, data-driven control and optimization;
11. modularity of production systems;
12. service-oriented computing architectures;
13. symbiotic interaction between humans and cyber-physical systems.

Among the presented items, in the author's view, an HMS application is aligned with the topics: 1, 2, 4, 8, and 11.

In brief, Industry 4.0 can be defined as a novel industrial model of communication and inter-connectivity in real-time of significant amounts of data between CPS and people. The aim is to accelerate the decision-making process, and systems adaptability (SCHUH *et al.*, 2017).

Additionally, Industry 4.0 is motivated by high dynamic market fluctuations (OLIVEIRA, 2019) and the emergence of a high number of connected resources throughout the whole manufacturing process. The connectivity targeted by the Industry 4.0 enables the emergence of flexible and reactive control systems based on the cooperation of entities during the decision-making process(DERIGENT *et al.*, 2020).

Considering the presented reactivity and connectivity required by Industry 4.0, the concept of decentralized production architectures and especially HMS experiences a new relevance (GRÄSSLER; PÖHLER, 2017). Besides the promised capabilities, the HMS presents an autonomous entity called holon as a basic unit. The conceptualization of a holon and a CPS have a degree of similarity and overlap of definitions (WANG; HAGHIGHI, 2016). Both concepts rely on integrating logical and possible physical parts, which interact, exchanging information between the digital and physical components to fulfill a specific objective.

3.2 Cyber-Physical Systems

In this subsection, the concepts of Cyber-Physical Systems (CPS) and Cyber-Physical Production Systems (CPPS) are summarized as their relevance is vital and inspires the development of this research work as a contribution to the Industry 4.0 domain.

Cyber-Physical Systems (CPS) are systems characterized by intense collaboration between their components, which are connected with the physical world, requiring and using services of data processing and accessing data available on the internet (MONOSTORI *et al.*, 2016). The key element in a CPS is the interaction between the cyber and the physical world: "CPS is about the intersection, not the union, of the physical and the cyber. It is not sufficient to separately understand the physical components and the computational components" (LEE; SESHIA, 2017). Practical examples of CPS realization can be elected: autonomous cars, robotic surgery, intelligent buildings, smart electric grid, and smart manufacturing (NIST, 2013).

The CPS emergence relies upon the development of computer science (CS) and information and communication technologies (ICT). The parallel development of ICT and CS led to a convergence between the cyber and physical world is depicted in the figure 5. The figure shows the HMS as an initiative to merge cyber and physical worlds, which started and the physical side.

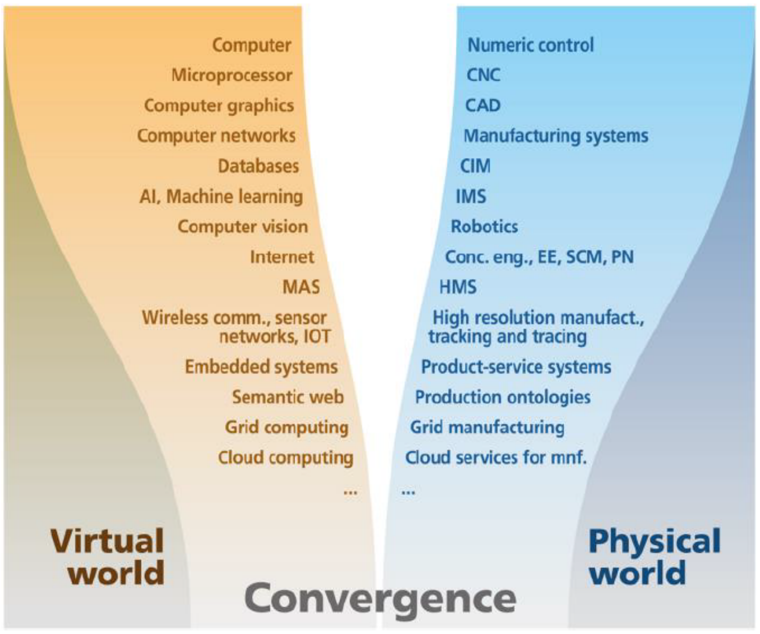


Figure 5 – Information and communication technologies convergence, adopted from (MONOSTORI *et al.*, 2016)

The origin of CPS is usually associated with the embedded systems (PARK *et al.*, 2012). Such systems consist of mechanical and electrical parts that perform specific functions within real-time computer constraints. Hence, the embedded systems are characterized by interactions between the computational features and the physical constraints of the physical world. The term CPS can be tracked back to 2006 when the first National Science Foundation (NSF) WorkShop on Cyber-Physical Systems was held in Texas, USA. In its first years of maturation as an official term, the CPS concept was associated as an enabling technology for various application fields (MONOSTORI *et al.*, 2016):

- agriculture;
- building controls;
- defense;
- energy response;
- energy;
- healthcare;
- manufacturing and industry;
- society;
- transportation.

In the German context, the National Academy of Science and Engineering (ACATECH) has provided a leading role in promoting the CPS concepts. The figure 6 represents a graphical representation of a maturity model developed in the Laboratory for Machine

Tools and Production Engineering (WZL) of RWTH Aachen University (MONOSTORI *et al.*, 2016).

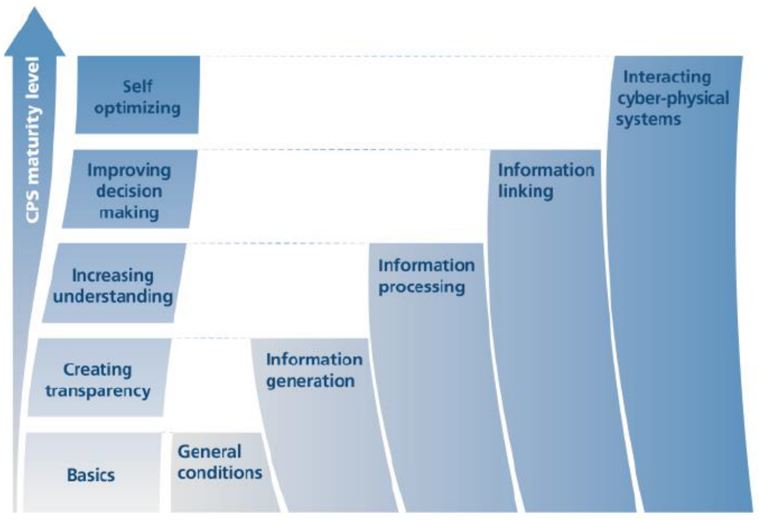


Figure 6 – CPS maturity level model, adopted from (MONOSTORI *et al.*, 2016)

Following this maturity model, on the first level (Basics), the organization and structural conditions for a CPS are created. In contrast, in the next four, higher maturity levels are represented by the information and knowledge processing and the cooperation, and collaboration aspects (MONOSTORI *et al.*, 2016).

Cyber-Physical Production Systems (CPPS) extends the primary idea of CPS to the manufacturing domain and may be considered as a pivotal basis for the 4th Industrial Revolution. CPPS consists of autonomous and cooperative elements connected across all production levels, from processes through machines to production and logistics networks. The three main characteristics of CPPS are highlighted here (MONOSTORI *et al.*, 2016):

- intelligence;
- connectedness;
- responsiveness.

The concept of CPPS works to break the traditional automation pyramid. The typical control and field levels still exist, for example, in the form of Programmable Computer Logic (PCL). However, the higher levels of the traditional pyramid are frequently related to more decentralized solutions. Figure 7 represents this breakdown (MONOSTORI *et al.*, 2016).

In summary, the CPPS is represented by two key components. The components that stay at the lower level are responsible for real-time functions, while the components in the higher levels have the tasks of performing data analysis, data processing, and intelligent decision-making (MONOSTORI *et al.*, 2016).

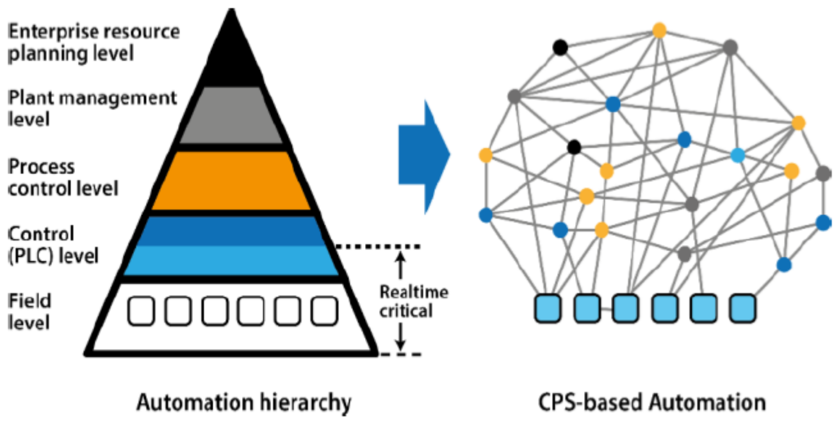


Figure 7 – Traditional automation pyramid breakdown, adopted from (VDI, 2013)

Figure 8 represents a architecture for developing a CPPS. From the data acquisition to the final level creation. Level 1 (connection level) represents the physical space, levels II-IV represent the cyberspace domain, and V represents the interaction between the cyber and physical worlds (MONOSTORI *et al.*, 2016).

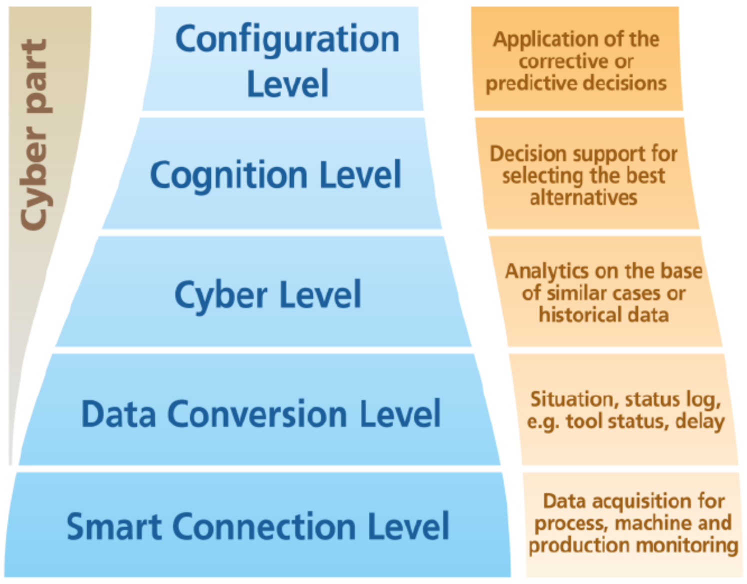

Cyber part
Configuration Level
Cognition Level
Cyber Level
Data Conversion Level
Smart Connection Level
Application of the corrective or predictive decisions
Decision support for selecting the best alternatives
Analytics on the base of similar cases or historical data
Situation, status log, e.g. tool status, delay
Data acquisition for process, machine and production monitoring

Figure 8 – Traditional automation pyramid breakdown, adopted from (LEE *et al.*, 2015)

3.3 Holonic manufacturing systems

The holonic manufacturing systems (HMS) may offer a suitable alternative to achieve the LMAS implementation requirements. The HMS brings to the manufacturing field the concepts elaborate by Arthur Koestler for living organisms and social organizations (KOESTLER, 1968). The HMS is based on the interaction of individual entities called holons. As Koestler explained, a holon is an identifiable part of a system that represents a group of functionalities, but at the same time, this part may also be a set of other parts (holons). In brief, a holon is a part of a whole and a whole for its subparts (holons) at the same time (LEITÃO; RESTIVO, 2006).

In the HMS perspective, a holon is an autonomous and cooperative part of a manufacturing system that works transforming, transporting, storing, validating information and physical parts. The holon may consist of both physical (optional) and logical components (mandatory). A holarchy is a set of holons following defined rules to achieve a production objective (BRUSSEL *et al.*, 1998).

3.3.1 Holonic manufacturing systems history

The HMS concept was created after a collaboration between the public institutes and private companies of USA, Japan, Australia, Canada and the European Community (EC), and the European Free Trade Association (EFTA). This international effort intended to start a two-year feasibility study on IMS. The program was divided into six different consortia, one of which, driven by 31 international industry, academic, and research institute partners, was responsible for an extensive study over the application of HMS concept (CHRISTENSEN, 1994):

As a result of this extensive study, an initial proposal for HMS was developed and formalized by Christensen: on that work, an initial architecture, system engineering methodology, and standardization directions for upcoming technologies and enabling technologies for HMS were proposed (CHRISTENSEN, 1994).

The HMS initial proposal considered the agile 21st-century manufacturing enterprise requirements. To cite a few, the 21st-century manufacturing enterprises should (CHRISTENSEN, 1994):

- bring out new products quickly;
- assimilate new experience and technological innovations easily continuously modifying its products accordingly;
- develop products to evolve;
- implement re-programmable, re-configurable, continuously changeable production systems with an integrated approach making the lot size irrelevant

A HMS consists of holons interacting and respecting a distributed control model to accomplish manufacturing goals. Due to its autonomous behavior and intelligent capabilities, the HMS concept promises to add an alternative to novel manufacturing paradigms to increase modularity, decentralization, autonomy, and scalability (LEITÃO; RESTIVO, 2006), increasing the responsiveness to market demands and technical disruptions. The generic approach of HMS control systems is reinvestigated in the context of Industry 4.0.

Further, it seems to be a suitable approach for LMAS operations planning and control and is going to be investigated in the context of this work in the section 3.3.

3.3.2 Holonic manufacturing system architectures

In consonance with the efforts to increase HMS's flexibility and adaptability, architectures have been developed PROSA (BRUSSEL *et al.*, 1998), ADACOR (LEITÃO; RESTIVO, 2006), and ADACOR² (BARBOSA *et al.*, 2015).

The table 1 summarizes studied HMS architectures which served as a basis for the design of this research work along with the classification of the architecture defined on (DERIGENT *et al.*, 2020): the historical architectures represent former proposals frequently cited and used as a basis for the recent holonic architectures; the dynamic represents the architectures focused on adaptability and system reactivity; the web-oriented refer to the architectures for cloud-based solutions.

Table 1 – Holonic manufacturing systems architectures classifications

Architecture	Architecture classification
PROSA (BRUSSEL *et al.*, 1998)	Historical
ADACOR (LEITÃO; RESTIVO, 2006)	Historical
HCBA (CHIRN; MCFARLANE, 2000)	Historical
PROSIS (PUJO *et al.*, 2009)	Historical
ORCA (PACH *et al.*, 2014)	Dynamic
(BORANGIU *et al.*, 2015)	Dynamic
ADACOR² (BARBOSA *et al.*, 2015)	Dynamic
POLLUX (JIMENEZ *et al.*, 2017)	Dynamic
(QUINTANILLA *et al.*, 2016)	Web-oriented

3.4 ANEMONA methodology

Various methodologies containing engineering design procedures have been devised as a guide for HMS development and Multi-Agent systems (MAS): The VDI2206 (GAUSEMEIER; MOEHRINGER, 2002) presents a flexible procedure model for the development of mechatronic systems proposing the V-model and process modules to be used on mechatronic system development steps, though not focussing HMS or MAS in particular; Designing Agent-based Control Systems (DACS) methodology is probably the first engineering approach for agent-based control systems, focusing on a semiautonomous collaborative method (BUSSMANN *et al.*, 2004); ROMAS (GARCIA *et al.*, 2014) addresses the engineering of normative open systems using the multi-agent paradigm, presenting a multi-agent architecture, meta-model, methodology, and Computer-Aided Software Engineering (CASE) tool. In ANEMONA methodology's (BOTTI; GIRET, 2008) the aggregation feature to the agent definition is included suitable for HMS applications.

The HCA presented in this work is built with the support of the ANEMONA methodology. It explicitly declares the concept of holons over the agent conceptualization. In the ANEMONA methodology, the authors complement the agent definition to the so-called Abstract Agent to be used as the basic unit for the system design. The Abstract Agent can be a unique, indivisible agent or a group of agents. With that definition, the authors benefit from the agent technology to implement a holonic manufacturing system while keeping the conceptualization compatible with the holon's former definition (BOTTI; GIRET, 2008). ANEMONA comprises the five phases: analysis, design, implementation, setup, and operation/maintenance, represented by figure 9.

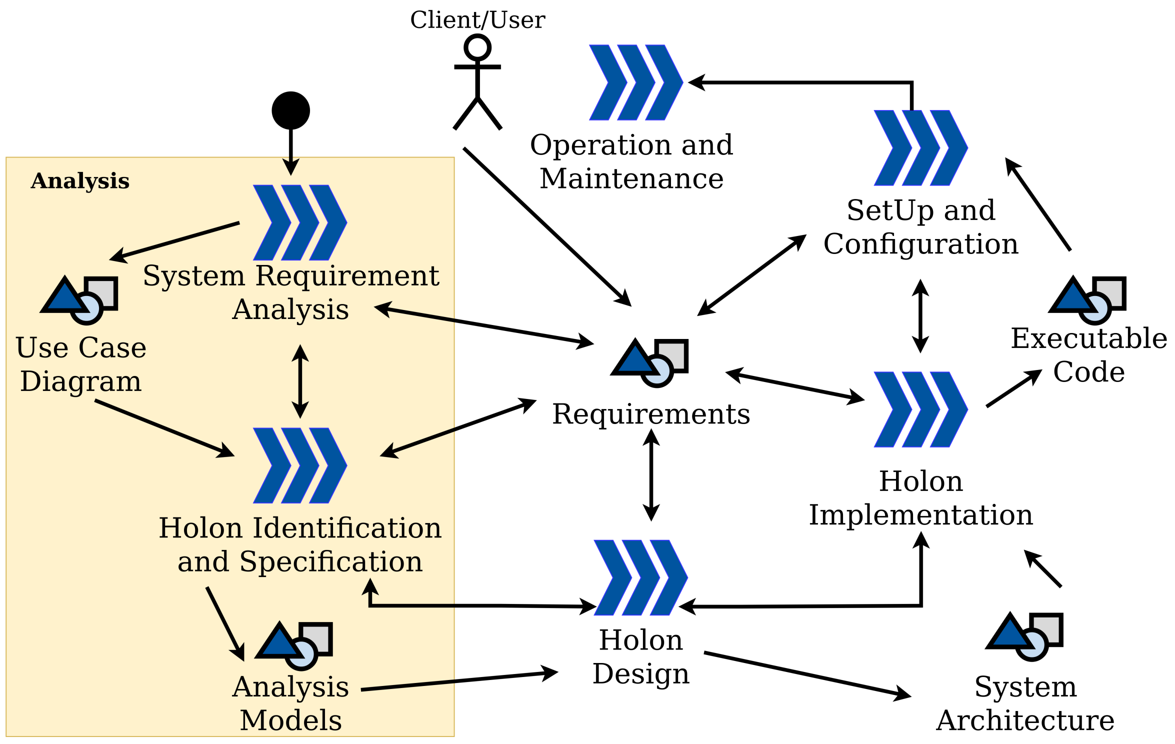


Figure 9 – ANEMONA development phases (BOTTI; GIRET, 2008)

The ANEMONA methodology is chosen among other multi-agent development methodologies due to its bound to the holonic concept, primarily when referring to differences between the agent technologies and the HMS concepts, highlighting how an HCA can be designed to benefit from the HMS concepts. ANEMONA methodology defines an iterative process over design, analysis, and implementation phases. Following the ANEMONA methodology, researchers develop use cases and models containing the holon objectives, roles, and behavior descriptions. The list below presents the ANEMONA system architecture models (BOTTI; GIRET, 2008):

- Organization model: this model is used to specify the main architectural elements of the HMS specifying the relationships between the main entities;
- Interaction model: this model is used to determine the dynamic behavior of the HMS. It is composed of abstract agents, agents, roles, goals, abstract goals, tasks, abstract tasks, interactions, and interaction units;
- Agent model: the agent model specifies the abstract agents' autonomy, intelligence, and conceptualization. The agent model is a set of agent diagrams. Each agent diagram might contain goals, beliefs, and tasks.
- Task and goals model: this model is used to specify the consequences of tasks present in the organization, interaction, and agent models;
- Environmental model: the environment model defines the environment entities the HMS holons may interact with. It is focused on the abstract agent's perceptions and actions.

In chapter 5 a subsection of the ANEMONA system architecture models is presented. The novel LMAS paradigm briefly presented in the Introduction chapter may find a suitable architectural realization to fulfill its basic requirements due to the HMS's autonomy, cooperative and reactivity characteristics.

3.4.1 ANEMONA notation

For comprehension reasons, it is essential to present the ANEMONA notation. The ANEMONA, notation, figure 10, presents a collection of symbols that represent the agent properties and interactions. This notation is used in the interaction, environment, tasks and goals, Agent and organization models. The description of the notation is listed below, (BOTTI; GIRET, 2008):

- **Abstract agent**: an autonomous entity that is abstract and complex. An abstract agent represents non-atomic holons that are in turn composed of holons;
- **Agent**: an autonomous, reactive, proactive, and social computational entity, which can act in an environment. In an HMS, an agent may represent atomic holons;
- **Interaction unit**: t can be a message or an event. An interaction unit details who executes it (the abstract agent that initiates it), who collaborate with it (the abstract agent to which the interaction unit is sent), and what tasks are executed in the interaction unit;

- **Abstract task**: is used to represent an abstract agent capability. It can be a task (when the abstract agent is a single agent);
- **Task**: it is used to model the capabilities of an agent. A task represents the functionality of the agent. An agent can modify its environment employing tasks;
- **Abstract belief**: it is used to model the belief of an abstract agent. It may be decomposed into agents' beliefs or group beliefs;
- **Belief**: it is the mental entity that models the idea that an agent has about its surroundings;
- **Abstract goal**: it is used to represent the goal of an abstract agent (non-atomic holon);
- **Goal**: it describes what an agent (atomic holon) is trying to fulfill, what it is looking for, the reason for its execution;
- **Organization model**: represents a group of abstract agents which cooperate to achieve common goals;
- **Interaction**: it is used to specify dependencies among abstract agents and to define their behavior. An interaction shows what the reaction of an abstract agent is to a given event, message, environment status;
- **Abstract**: information structure: it represents a collection of abstract agent beliefs;
- **Information structure**: it represents a collection of agent beliefs;
- **Event**: it models changes in the environment that an abstract agent perceived.

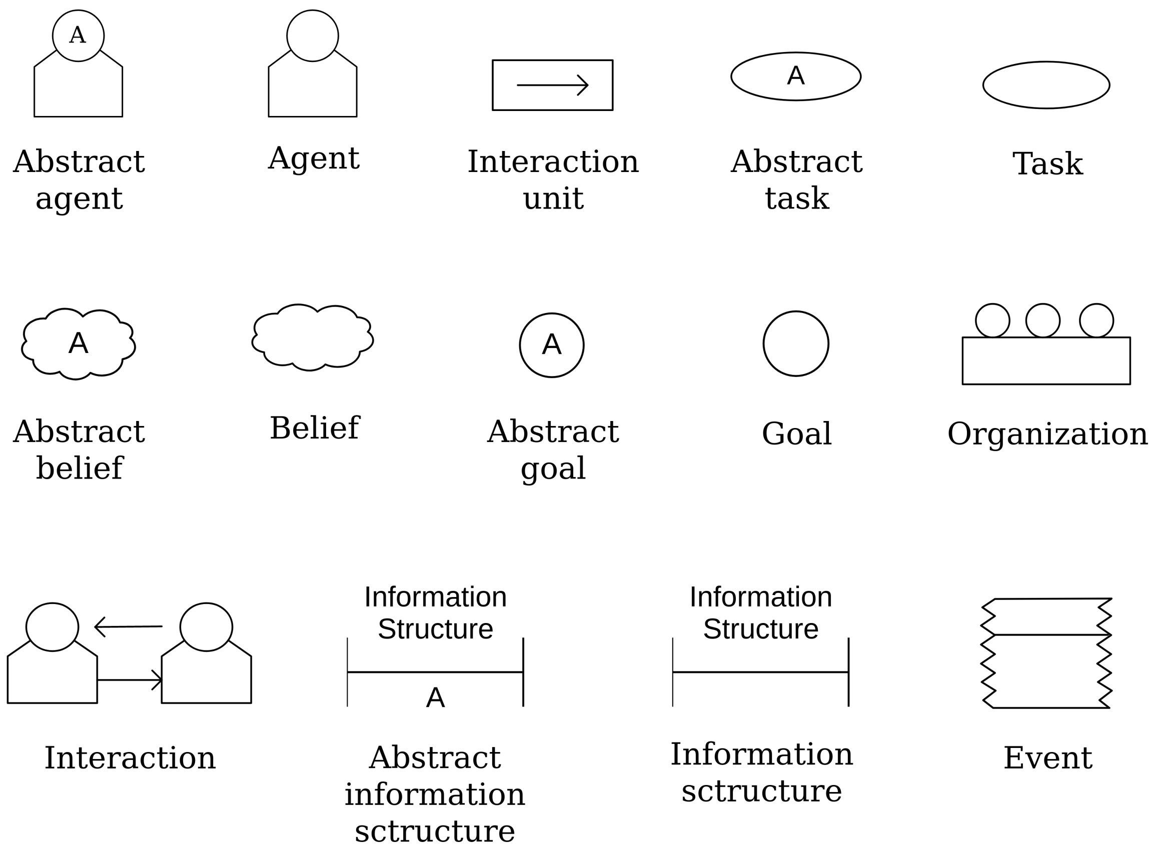


Figure 10 – ANEMONA notation, adopted from (BOTTI; GIRET, 2008)

3.5 Multi-Agent Systems

The multi-agent systems (MAS) study began about 40 years ago, related to distributed artificial intelligence (DAI). DAI is a sub-field research area of artificial intelligence. DAI field of study concerns how a group of modules cooperates to divide and share a task. It researches the techniques necessary for coordination and actions management in a multi-agent environment but considers the agents with a fixed behavior model (GIRET; BOTTI, 2004).

MAS studies the coordination of intelligent behavior of a group of autonomous intelligent agents. It focuses on individual behavior and mainly on behavior models, cooperation and coordination strategies, intelligent brokerage, performance optimization, learning from experiences, and coalition formation. The MAS can be understood as a software technology inspired by the requirements of autonomy, cooperation, and group formation. Due to that characteristics, MAS is applicable for an extensive range of domains. In the manufacturing domain, to cite a few: real-time control, planning and scheduling, enterprise integration and supply chain management, and product design (GIRET; BOTTI, 2004).

The agents which compose a MAS are autonomous and flexible computational systems acting in a specific environment (WOOLDRIDGE; JENNINGS, 1995). The usual properties of MAS are (FRANKLIN; GRAESSER, 1997; NWANA, 1996):

- autonomy: agents can operate without human intervention;
- social ability: agents communicate with others to fulfill their objectives;
- rationality: agents reason about collect data to find an optimal solution;
- reactivity: agents detect the environment changes and react accordingly;
- pro-activeness: agents are capable of acting guided by their objectives and goals;
- adaptability: the agent can learn with the experiences;
- mobility: agents can move through a network;
- veracity: agents do not provide false information;
- benevolence: agents help each other in order to fulfill their goals unless the other agents act against their own goals.

3.6 Robot operating system

The Robot operating system (ROS) is an alternative for the ANEMONA implementation phase preferred development framework (JADE). It is not an agent-dedicated platform; however, it provides all the functionalities to establish communication among entities and benefits related to code-reuse for robotic applications.

ROS is an operating system for robots that provides hardware abstraction, low-level device control, implementation of common-used functionalities, message-exchanging between software entities, and packages management. ROS provides the possibility to build, write and distribute software solutions for robots on various computer platforms.

One of ROS's main characteristics is code reuse for developing robotics applications. Several packages in the ROS universe contain robot 3D models, dynamic motion controllers, navigation, mapping, and image processing algorithms commonly applied for robot operations (e.g., screw driving, pick & place).

The code reuse accelerates the development time, providing quick prototyping, results, and feedback. However, it is possible to write specific packages to implement customized solutions. In 2007, the ROS developers community counted with more than 2477 authors, 181509 code contributions (commits), and 14 million lines of code. The ROS community encourages and provides tutorials teaching how to write packages for the ROS correctly.

ROS provides communication mechanisms in different patterns: asynchronous communication via topics, synchronous communication via services and action services, or parameter server.

3.6.1 Asynchronous messages

In this approach, messages are routed using a publish and subscribe semantic. A node sends a message publishing to a topic. The nodes that are interested in that information subscribe to this topic. Topic publisher and subscriber are not aware of the existence of each other. The idea is to decouple the producer from the consumer.

3.6.2 Synchronous messages

When a node sends a message and needs a response, topics are not a reliable option due to the unidirectional nature of its implementation. In this case, ROS services utilize a bi-directional communication mechanism between two nodes. A node can send a message directly to another node and wait for the answer with ROS services. A variation of ROS services is the ROS action servers. The ROS action services provide a dynamic feedback message stream while the server node executes the service request. The client waits for the server without a feedback stream in the normal ROS server.

3.6.3 Parameter server

The parameter server is a center of information, where nodes can set or get parameters values. The parameter server is practically used in all ROS applications, especially for publishing and getting constants or input control parameters. The parameter server is shared among all the ROS nodes in the network, making the information published to it unsafe.

3.7 Line-Less Mobile Assembly System

The Line-Less Mobile Assembly Systems (LMAS) is an assembly paradigm designed to address the lote size 1 large product assembly. It derives from the necessity of the flexibility increase regarding the mobility of products and resources, not addressed at this level on former paradigms like Flexible Assembly/Manufacturing Systems (FAS/FMS) and Reconfigurable Assembly/Manufacturing Systems (RAS/RMS) (BUCKHORST *et al.*, 2019). The LMAS paradigm relies on three main key principles:

1. **Clean Shop Floor**: the factory design should offer the few space constraints as possible. With that principle, the resource operations assignments carry a low chance to lead the whole system to a deadlock state. In a deadlock state, the movement of resources would be impeded, or the system would not be able to continue the assembly planning;
2. **Mobilization of all relevant assembly resources** this principle considers the mobilization of all product resources, including product, parts, industrial robots, and metrology systems, to cite a few;

3. **Unrestricted Assignment**: the last principle covers the fact that it should be possible to associate an assembly operation to a resource (e.g., transportation operation for an automated guided vehicle(AGV)) at a specific place and time on the shop floor. It is mandatory to meet the two previous principles to accomplish the third (BUCKHORST *et al.*, 2020).

The principles and the shop floor flexible organization, based on multi-purpose stations, for the LMAS, are presented in the figure 11. The factory configuration represents how the factory is organized to allocate the multi-purpose stations. The stations represent the entities capable of performing manufacturing operations. Each station's capabilities result from a combination of resources (e.g., AGV, tools, assembly robots, and operators), which variety of combinations give the LMAS paradigm great flexibility to match different product requirements.

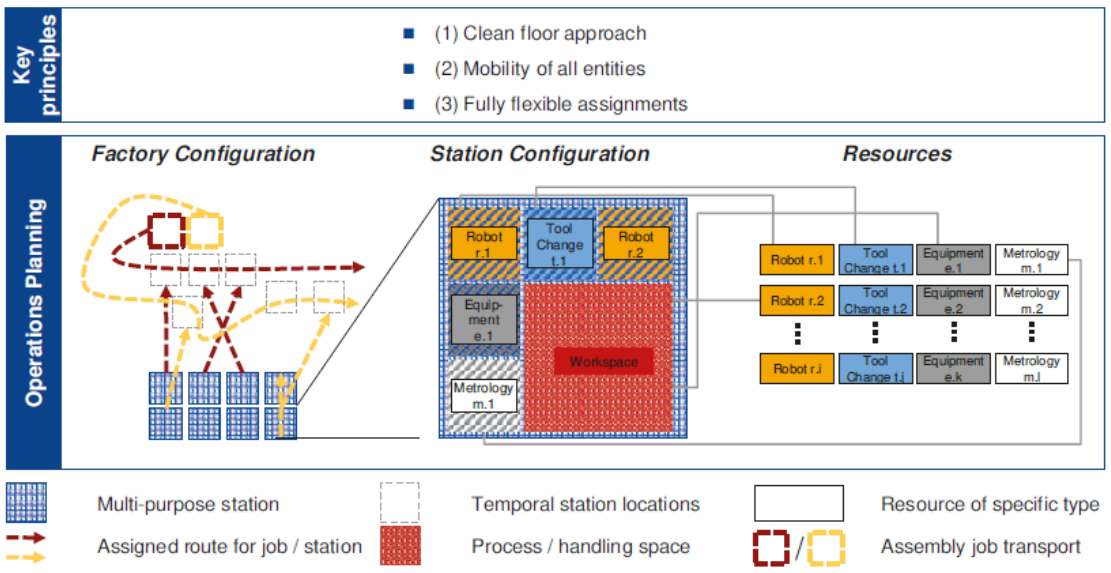


Figure 11 – Line-Less Mobile Assembly System: Key principles and operation planning organization, adopted from (BUCKHORST *et al.*, 2019)

The LMAS paradigm follows a semi-heterarchical approach. Depending on the system state (e.g., regular operation or disrupted mode), entities can negotiate autonomously or follow a strict optimal plan given by another entity in the hierarchy's upper position. This dynamic organization characteristic is one of the holons defined for the HMS, indicating that initially, the HMS is a promising approach to LMAS realization.

3.7.1 LMAS Architecture

As a design decision of former initiatives towards the definition of an LMAS architecture (BUCKHORST *et al.*, 2021, in Press) the LMAS paradigm architecture consists of the interaction between three layers: the planning, control, and execution layers.

The arrival of customer orders triggers the planning layer, and it is responsible for generating optimal temporary schedules for the assembly equipment present on the shop floor. The schedules contain assignments from assembly equipment to stations, operations sequences, and assembly equipment locations (BUCKHORST *et al.*, 2021, in Press).

The control layer is responsible for preparing the data to be processed by the optimization layer, interpreting the global schedule and distributing it through the available assembly equipment, requesting global schedules based on user or local (holon) input, and monitoring the system execution (BUCKHORST *et al.*, 2021, in Press).

The execution layer consists of the low-level holons, which autonomously and cooperatively execute the assembly process (e.g., assembly equipment holon) (BUCKHORST *et al.*, 2021, in Press). The low-level level holons are most likely to present a physical part since their use is related to the assembly operation itself.

3.7.2 LMAS Use Cases

As the current result of these former studies and analysis on the LMAS paradigm, UCs have been generated to help the modeling of the systems requirements and clarify directions regarding the system development (BUCKHORST *et al.*, 2021, in Press). This work relies on two UCs: Use case one (UC1) - Normal operation; Use case two (UC2) - Disruptions during normal operation.

UC1 is depicted in figure 12. It represents the intended workflow of the normal operation of an assembly system without any disruptions. In that case, the operator can insert a new customer order to initialize the optimization algorithm. The optimization algorithm generates an optimal schedule for all holons on the shop floor (e.g., stations, fixtures, testbeds), considering the shop floor's current state and the customer requirements. Once completed, the optimal plan is distributed to the holons through a control mechanism and executed considering the holon's capabilities. The optimal plan consists of customer orders executions sequence and the definition of where and when the resources should execute operations on the shop floor. The operator also can monitor the production, checking the progress of the overall system or individual aspects of production holons. The system should be prepared to accept new orders during the execution of an initialized customer order, reacting accordingly to the new shop floor state and customer demand.

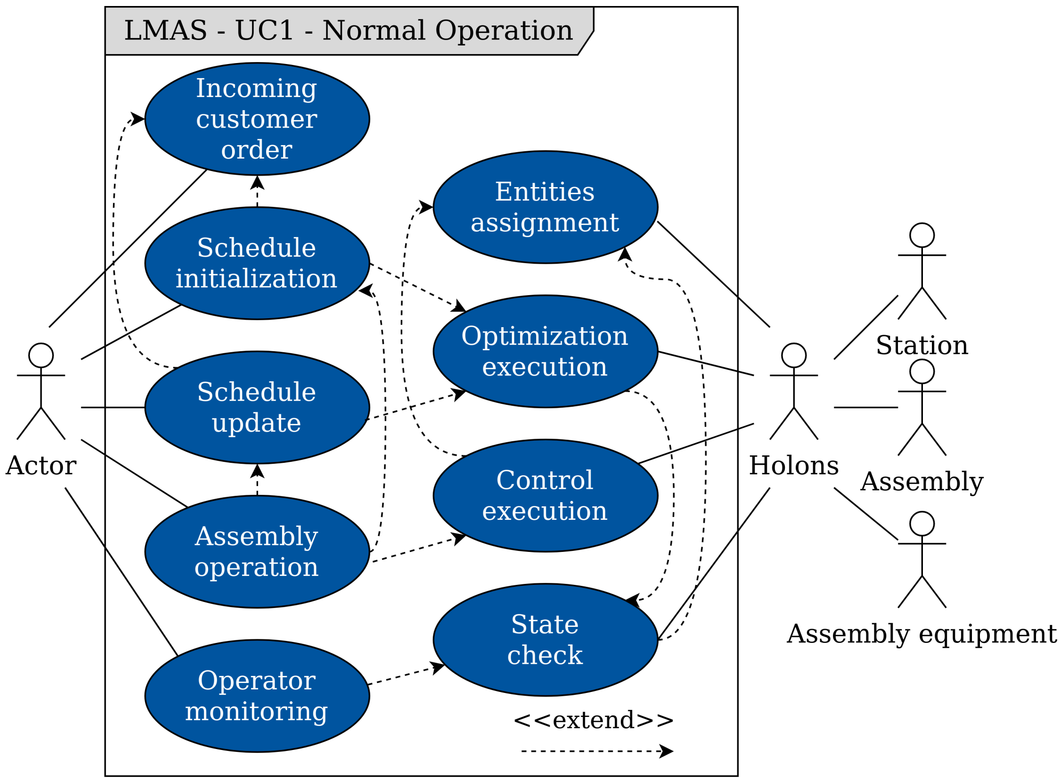


Figure 12 – Line-Less Mobile Assembly Systems UC1: normal operation, adapted from (BUCKHORST *et al.*, 2021, in Press)

UC2 is shown in figure 13. This use case represents how the production system should react to disturbances during the operation. For example, the disturbances may involve the absence of parts, assembly equipment, and station. After detecting these disturbances, the system shall perform a disturbance handling procedure to decide which operations should be executed to repair the detected fault. The consequence of the fault could lead the system to assume a compensation behavior or even perform a total system reconfiguration. In the last case, the system requires a new schedule; therefore, the optimization algorithm must be rerun to find a new optimal plan. Considering that manufacturing failure management is an open problem, a suitable heuristic to deal with the UC2 shall be chosen. Regardless of the chosen option to compensate for the disturbance, the system will check the holons' state, assign new locations, and control the execution necessary to realize the new configuration.

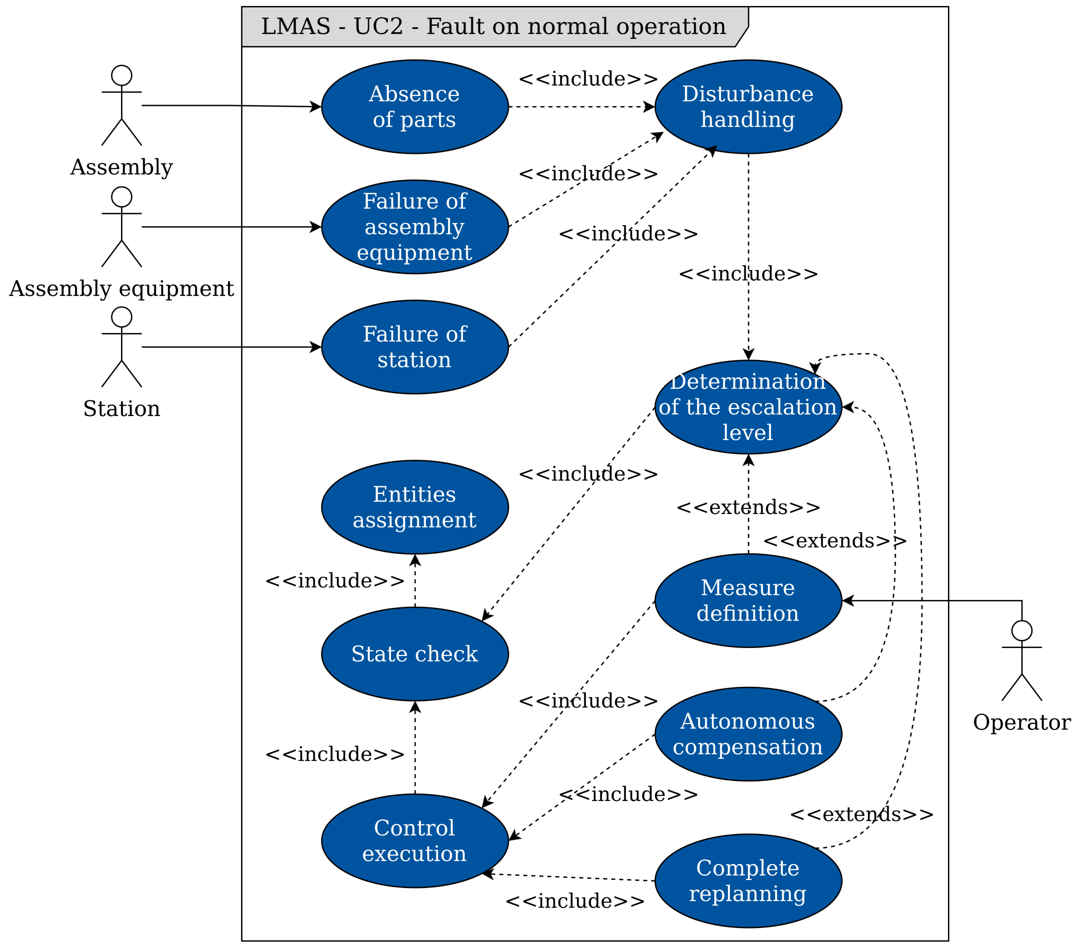


Figure 13 – Line-Less Mobile Assembly Systems UC2: disturbance during normal operation

3.7.3 LMAS Requirements

Envisioning to fulfill the requirements imposed by the UC1, UC2, and LMAS former contributions ((BUCKHORST *et al.*, 2021, in Press)), the proposed control architecture shall present the elements depicted in the table 2. The requirements are presented in four categories: process, product, process, and design patterns. The process, product, and process categories are inspired in the PROSA (BRUSSEL *et al.*, 1998) high-level knowledge exchange. The design patterns are requirements specified due to the experience and requirements of previous contributions to LMAS.

Table 2 – LMAS requirements.

	ID	Requirements
Resource	RQ-01	Multipurpose station and resources
	RQ-02	Fabric model / Shop floor management
	RQ-03	Mobile resources / temporary station locations
	RQ-04	Spatio-temporal order relations between stations and resources
Product	RQ-05	Products, parts and assemblies structures (physical relationships)
	RQ-06	Products, parts and assemblies manufacturing data and information
Process	RQ-07	Individual order routes
	RQ-08	Interface to transportation services
Design patterns	RQ-09	Global manufacturing plan
	RQ-10	Strategies for delay and failure compensation
	RQ-11	Central control / Central monitoring
	RQ-12	Decentralized order execution
	RQ-13	Ontology-based
	RQ-14	Type/Instance

3.8 Ontology

Realizing an HMS for the LMAS paradigm involves the dynamic creation of complex cooperative communication structures between the basic entities (holons) depending on the system state. During the structure creation process, the holons should understand and have the necessary knowledge to interact with one another and their environment following specific objectives and behavior rules.

Besides, care must be taken to avoid the loss of semantic meaning between message exchanges. Different protocol usage is inevitable due to the connectivity of different entities and the heterogeneity of applications, resulting in message transformations and wrappers to communicate via the systems' interfaces.

Thus it becomes necessary to implement manufacturing ontologies to express clearly, differentiate, and precisely detail the system knowledge descriptions (e.g., environment, holons, objectives) (SIMÓN-MARMOLEJO; RAMOS-VELASCO, 2018). Briefly, the ontologies carry the system concepts and vocabulary associated with a specific domain. In this case, the LMAS is a highly flexible manufacturing paradigm that concentrates on the flexible manufacturing domain. Figure 14 describes an example that shows a section of a manufacturing ontology.

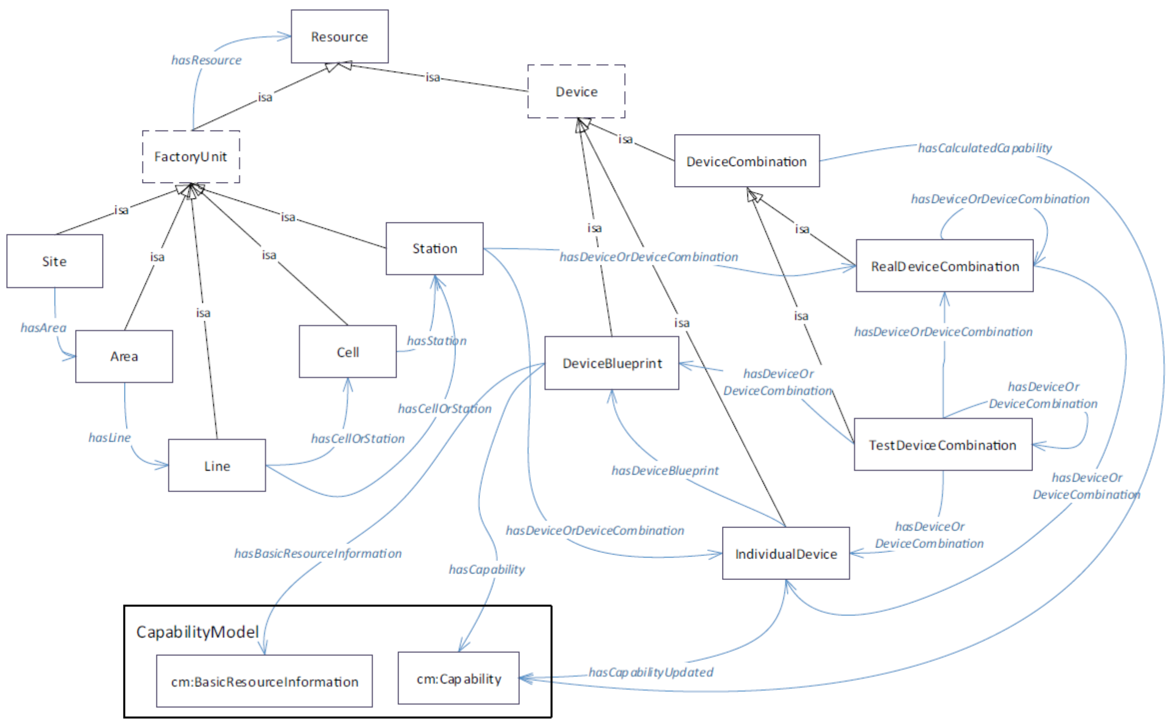


Figure 14 – MaRCO Ontology sub-section, adopted from (JÄRVENPÄÄ *et al.*, 2019)

Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. The enabling description languages for Semantic Web are: Resource Description Framework (RDF), Protocol and RDF Query Language (SPARQL), Web Ontology Language (OWL), and Simple Knowledge Organization System (SKOS) (W3C, 2015).

The term ontology comes from philosophy and means the study of the being. It works like an artifact able to describe categories and relationships of all existing things (GUARINO, 1995). Typically, for engineering applications, "all existing things" is limited to a context of interest, and the ontology about this context is described as an explicit specification of the context concepts. When shared, the ontology is an engineering artifact that computational entities can use to visualize and understand the world in which they are inserted (OBITKO *et al.*, 2010). For example, an extrusion machine uses the ontology to execute a specific operation or communicate with other entities, interpreting the extrusion domain's terminology.

The context definition of an ontology is usually defined as a formal description and consists of concepts representing classes of objects in the real world, attributes, relations, and constraints. The ontology works as a static vocabulary describing a particular reality. (OBITKO *et al.*, 2010). The realization of an ontology is composed of instances of the ontology classes, usually called the knowledge base. In this work, the instantiation of an ontology for a specific domain is named a data model.

4 Literature review

The literature review covers two essential aspects of this research project: the HMS architectures and the manufacturing field's ontologies. Among other subjects to be researched, ontologies are chosen because this dissertation's proposal is also to (along with the holonic architecture models) implement a data model as an instantiation of a high-level manufacturing domain ontology. It is intended to research and find out ontologies that provide sufficient expression power to represent the holonic and manufacturing concepts essential to connect the HMS concept to the LMAS principles. In case of a lack of a ready-to-use ontology, an ontology will be proposed based on the research findings. The methodology used applied consultation utilizing specific keywords in scientific research repositories.

The actual research explored in detail 38 works related to HCA implementations and respective methodologies and 13 other scientific contributions to the manufacturing ontology field. A subset of 9 HCAs implementations and three ontologies contributions from this collection of works is described below. The Citavi® software is being used to organize the research literature. It is foreseen for the subsequent phases of the project to summarize other works of great relevance for the development of this project.

This work's keywords are Holonic Manufacturing System, Line-Less Mobile Assembly Systems, and Holonic Control Architecture.

4.1 PROSA: Product, Resource, Order and Staff architecture

Title: **Reference architecture for holonic manufacturing systems: PROSA**.

One of the more consistent holonic manufacturing architectures is the Product Resource Order Staff Architecture (PROSA) (BARBOSA *et al.*, 2015). The PROSA architecture relies on four different types of holons: (1) resource holon: it consists of a production resource and holds the operations knowledge necessary to drive production (e.g., operation times, waiting times, resource capabilities); (2) product holon: it holds the processes and product knowledge in order to achieve the correct product manufacturing; (3) order holon: represents a task on the shop floor, it is responsible for resource assignment, operations control and production monitoring; (4) staff holon: this holon is designed to be a multi-purpose holon in order to add specialized functionalities to the production system, such as optimization (BRUSSEL *et al.*, 1998).

The PROSA architecture also defines the high-level knowledge exchanged between the holons and a behavioral interaction model, which describes the holons in preparation and production execution. The holons exchange production (order-product holons), process (resource-product holons), and process execution (order-resources holon) knowledge in order to drive the manufacturing processes, see figure 15. The behavioral interaction model is described for three different situations: launch of a new order, addition of a new resource, and the order driving production (BRUSSEL *et al.*, 1998).

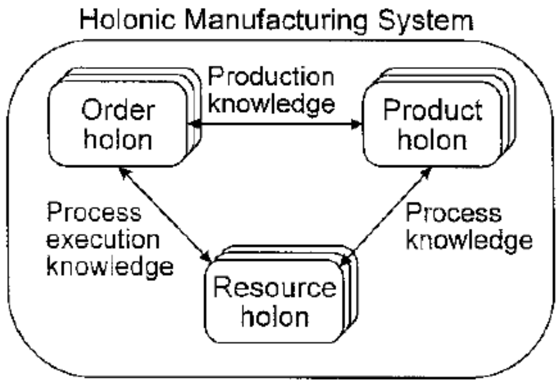


Figure 15 – PROSA: holon's knowledge exchange, adopted from (BRUSSEL *et al.*, 1998)

The PROSA architecture is a seminal work for an enormous amount of holonic architectures researched in the context of this work. One reason may be its high-level definition of Product, Resource, Order, and Staff holons which can be specialized to more complex and application-oriented holons. In the context of this dissertation, the PROSA architecture holons will be used as a basis for the application-oriented holons. This choice is justified as the PROSA offers an abstract definition aligned with the field of application of this work.

However, the PROSA architecture does not define complex holon data, interactions, states, and behavior as its conceptualization is bound to a high-level system definition. This dissertation aims to use PROSA as a starting point and define the necessary interactions over its seminal holon's definition.

4.2 ADACOR: ADAptative holonic COntrol aRchitecture

Title: **ADACOR: A holonic architecture for agile and adaptive manufacturing control**.

A more recent architecture called ADAptative holonic COntrol aRchitecture (ADACOR) also addresses the HMS. The ADACOR is similar to PROSA in defining the basic holons. ADACOR addresses four types of holons: product (PH), operational task (OH), task holons (TH), and supervisor holons (SH). The PH, OH, and TH are similar to the PROSA holons, whilst the SH is a holon specialized in developing optimal schedule assignment and control holon groups (LEITÃO; RESTIVO, 2006). The authors of ADACOR claim that the SH is different from the PROSA staff holon regarding the latter considers the coordination and the optimally of resource assignment. The figure 16 represents the ADACOR holon's organization.

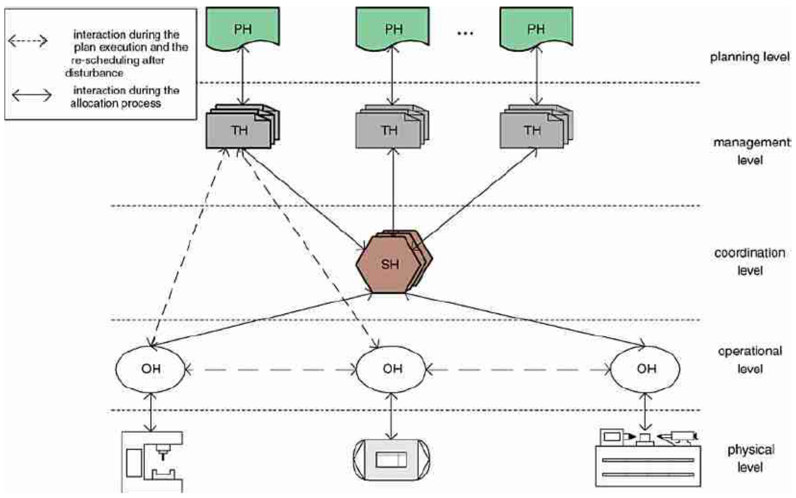

interaction during the plan execution and the re-scheduling after disturbance
interaction during the allocation process
PH
PH
...
PH
planning level
TH
TH
TH
management level
SH
coordination level
OH
OH
OH
operational level
physical level

Figure 16 – ADACOR: Holon's organization, adopted from (LEITÃO; RESTIVO, 2006)

Regarding the behavioral model, the ADACOR presents a mechanism to drive normal production scenarios and support disruptions during the execution phase. The system is designed to change between two states: stationary and transient. First, the system takes the form of a hierarchical system, following optimal schedules as advice given by the SH. The optimal schedule results from the interaction between various instances of the PH, TH, OP, and SH. When a fault occurs (e.g., resource fails), the system enters the transient state, activating the heterarchical mode. Each holon has the so-called autonomous factor that indicates the level of autonomy given to a specific holon during a fault. The holon, on which the failure occurred, spread a pheromone to the holonic neighborhood. When interacting with this pheromone, each holon has its level of autonomy increased. The intensity of the pheromone falls according to the layer distance between the holons. At a certain level of the autonomous factor, the holon stops following the optimal plan and interacts directly with the order holon to deal with the failure. When the pheromone losses its influence, the holons return to the heterarchical model (LEITÃO; RESTIVO, 2006). The figure 17 represents the changing of states of ADACOR.

The hierarchical and heterarchical modes in the ADACOR architecture aim to use the advantages of each organizational structure. The optimality is a complex task to achieve in distributed systems due to the restricted knowledge that each entity has, while in heterarchical mode, due to the centralized control, one entity has the entire system knowledge, and the optimality is easier to solve. At the same time, disruptions and unforeseen events cause less impact on distributed systems with autonomy, while on centralized systems, they can cause critical errors leading to a complete restructuring of the initial plan.

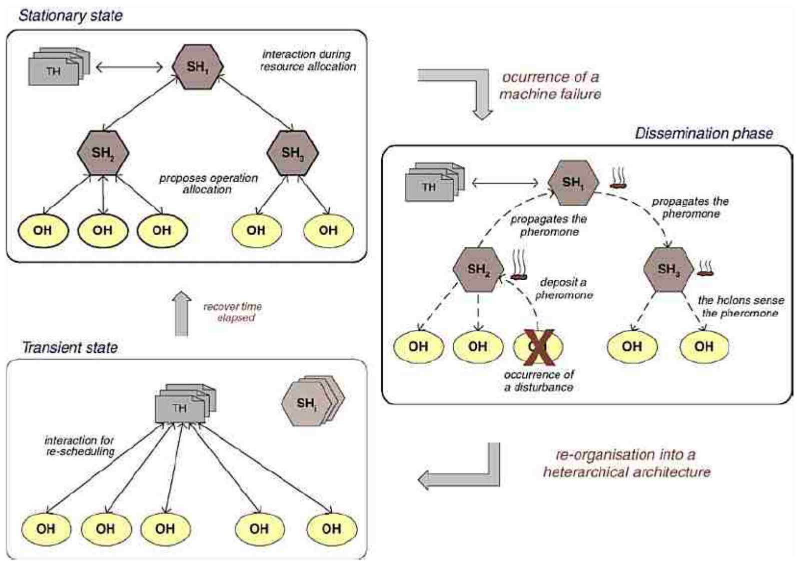

Stationary state
TH
SH1
interaction during resource allocation
SH2
SH3
proposes operation allocation
OH
ocurrence of a machine failure
Dissemination phase
propagates the pheromone
deposit a pheromone
the holons sense the pheromone
occurrence of a disturbance
recover time elapsed
Transient state
interaction for re-scheduling
re-organisation into a heterarchical architecture

Figure 17 – ADACOR: Stationary and Transient state, adopted from (LEITÃO; RESTIVO, 2006)

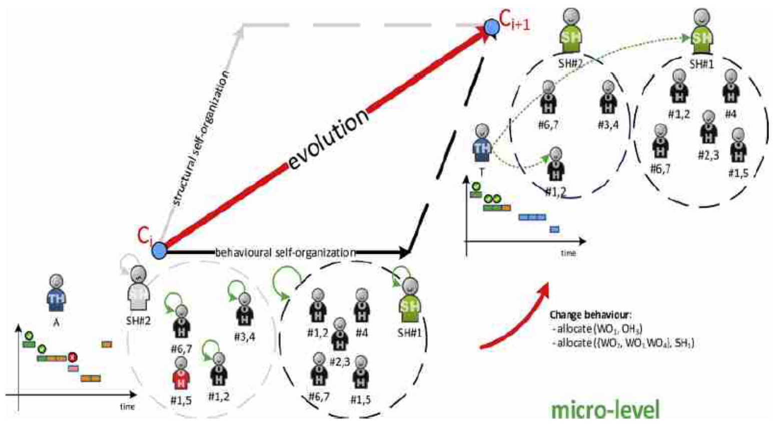
Though the ADACOR architecture introduces different holons than the PROSA, there exists a certain degree of similarity between the high-level functionality of each holon. The ADACOR is focused on the system's dynamic behavior and clarifies the difference between stationary and transient states given detailed interaction procedures during each phase. This dissertation will use the system state-changing mechanism presented on ADACOR but will not consider the pheromone as a strategy to inform the holons that a disturbance has occurred. This dissertation proposal also extends the definition of ADACOR holons introducing specializations not contained in the ADACOR architecture.

4.3 ADACOR^2: The evolution of ADACOR

Title: **Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution**.

The evolution of ADACOR (ADACOR^2) expands the former architecture adding a more complex treatment to the evolutionary states. ADACOR^2 utilizes the same concepts of ADACOR concerning the holons' structure. The ADACOR^2 considers aspects from the Darwinian Evolution theory, chaos theory, and biology to derive a concept of more than just two evolutionary states. The authors propose a system where the holons constantly look for opportunities to evolve. The evolution of the system may drive local micro changes (holon level) or even a more intense macro modification (e.g., rush order or new big order upcoming) (BARBOSA *et al.*, 2015). Figure 18 represents the evolution predicted by ADACOR^2, in this case of a micro-change which can lead to a self-organization behavioral modification or even a complete structural organization.

To detect the evolutionary opportunities, ADACOR^2 considers at the holon level a



evolution
structural self-organization
behavioural self-organization
C_i
C_{i+1}
Change behaviour:
- allocate(WO₁, OH₁)
- allocate({WO₁, WO₂WO₄}, SH₁)
micro-level
time

Figure 18 – ADACOR^2: States, adopted from (BARBOSA *et al.*, 2015)

learning mechanism that proposes insights based on the holon environment. The insights are evaluated by a reasoning module that assesses the efficiency of the proposals in a classic control-inspired mechanism. The reasoning module is designed to system stability regulation to avoid a possible chaotic behavior (BARBOSA *et al.*, 2015). Figure 19 represents the ADACOR^2's reasoning mechanism.

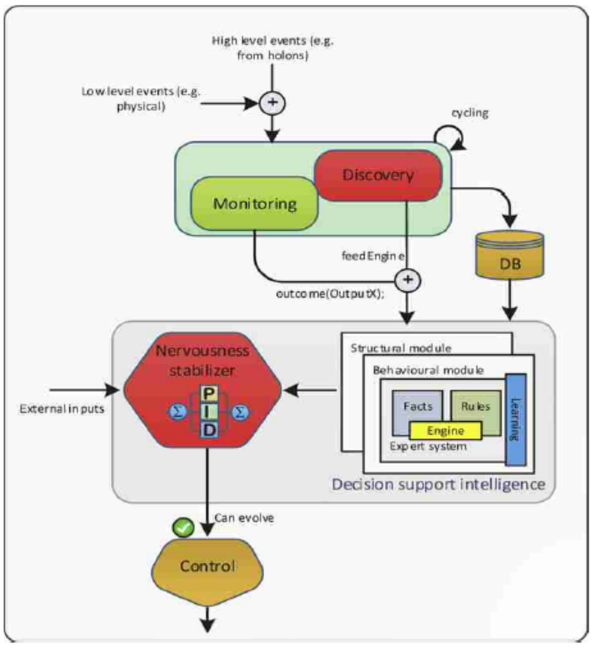


Figure 19 – ADACOR^2: Reasoning mechanism, adopted from (BARBOSA *et al.*, 2015)

The evolution of ADACOR proposes using a reasoning mechanism to evolve to a new state based on evolutionary opportunities. This dissertation will use the idea of the reasoning mechanism, which is intentionally very simplified, to deal with system disruptions such as resource faults, product defects, and rush orders. The development of a reasoning mechanism proposed by the ADACOR^2 proposal would be a whole dissertation due to the variety of solutions that may evolve to new states. The ADACOR^2 holons are built over the former ADACOR definition, and the same considerations are taken regarding the holonic definition.

4.4 Pollux

Title: **Pollux: a dynamic hybrid control architecture for flexible job shop systems**.

The Pollux architecture focuses on using governance rules in the "what-if" scenarios to modify the system state. The changes in the system behavior are triggered by disturbances observed by the control higher level. The opportunities to change followed by modifying the system's holons structural behavior are known as switches. In the presence of a switch, the system simulates several different consequences regarding the performance achieved. The higher-level controls perform the one which shows the best performances among them.

The atomic entity in Pollux architecture is called a decisional entity. The decisional entity is composed of: an entity objective, a decision-making technique, a parameter, governance parameters, a communication module, a data storage module, and an execution mechanism.

The remarkable characteristic of the decisional entities is the governance decision-making mechanism. This mechanism is an explicit set of parameters that define the rules of the holon's conduct. The governance rules can change, for example, objectives, interrelation with other entities, the decision-making technique, the roles of other entities; the priority of objectives; machine parameters; AGV's speed. The communication module gives the decisional entity features to transmit and understand other entities' messages. The data storage selects and stores relevant data during system operation. The execution module executes the actions of the decisional entity and is dependent on the entity's role. The architecture is composed of three basic decisional entities: local decisional entities (LDE), resource decisional entities (RDE), and global decisional entities(GDE). The decisional entity (a) and the decisional entities layer distribution (b) are depicted in the figure 20.

- The LDEs are located on the operation layer and are responsible for coordinating the online scheduling and jobs. The LDEs have all the manufacturing information related to the jobs. Their specific objective is to deal with the unexpected events that may occur during the system execution.
- The RDEs control the operation layer's resources (e.g., robots, conveyors, AGVs). They are responsible for providing manufacturing services that give the system the capability of producing the required products.
- The GDEs are located on the control layer and are responsible for the offline scheduling. The GDEs are related to the global objective and serve as the first

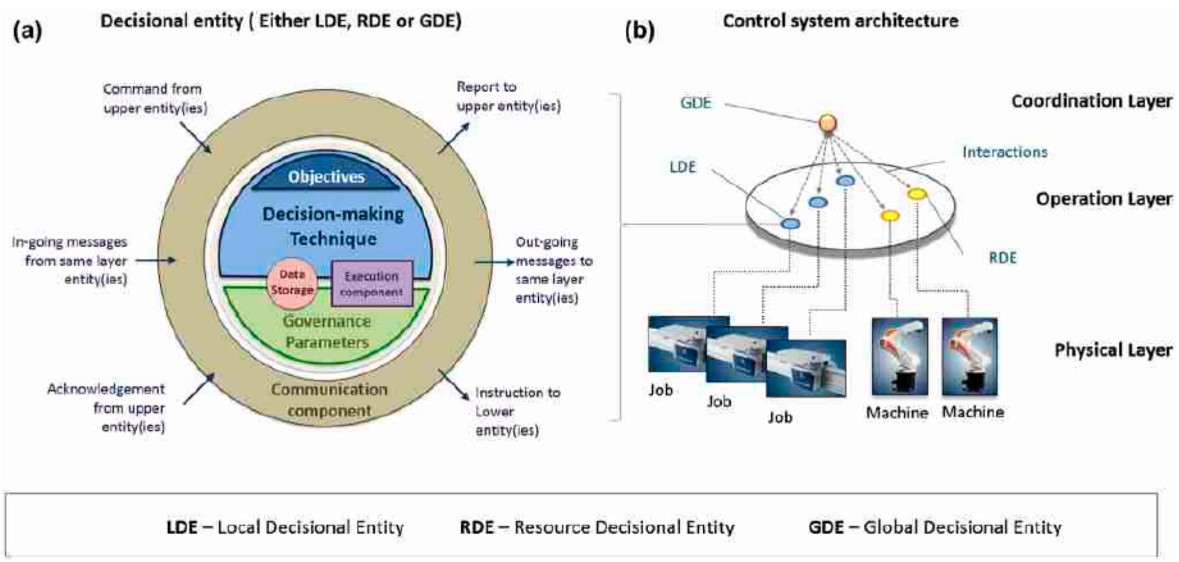

(a)
Decisional entity (Either LDE, RDE or GDE)
Command from upper entity(ies)
Report to upper entity(ies)
Objectives
Decision-making Technique
In-going messages from same layer entity(ies)
Data Storage
Execution component
Out-going messages to same layer entity(ies)
Governance Parameters
Acknowledgement from upper entity(ies)
Communication component
Instruction to Lower entity(ies)
(b)
Control system architecture
GDE
Coordination Layer
LDE
Interactions
Operation Layer
RDE
Physical Layer
Job
Job
Job
Machine
Machine
LDE – Local Decisional Entity
RDE – Resource Decisional Entity
GDE – Global Decisional Entity

Figure 20 – POLLUX: Decisional entities and layers description (JIMENEZ *et al.*, 2017)

optimal solution considering a system without disruption.

When the system is initialized, the GDE advises the LDE holons on scheduling the RDEs to drive the manufacturing production. In that first state, the LDE has low autonomy, so it follows the advice provided by the GDE. If an unexpected event is triggered (rush order, resource failures, product defect), the global entities are disconnected from the local decisional entities giving them sufficient autonomy to treat the failure based on the governance parameters (e,g, replace resource).

This dissertation will use the concept of the "what-if" rules presented in Pollux to deal with the system state transitions. The rules have to be researched and proposed in a simplified scenario. The Pollux presents the LDE entity responsible for the local optimization during a fault state. Unlike that approach, this dissertation pretends to split the adaptation tasks between collective holons that cooperate utilizing simplified rules.

4.5 Virtual Commissioning-Based Development and Implementation of a Service-Oriented Holonic Control for Retrofit Manufacturing Systems

Title: **Virtual Commissioning-Based Development and Implementation of a Service-Oriented Holonic Control for Retrofit Manufacturing Systems**.

In (QUINTANILLA *et al.*, 2016) is focused on the use of both Service-oriented Architecture (SoA) and HMS to contribute to the field of service-oriented holonic manufacturing systems (SoHMS). The overall architecture is supported by virtual commissioning, which tests the system using discrete-event simulation (DES).

An experimental platform was implemented using the SoHMS approach in the proposed system. The SoHMS system with the industrial equipment using TCP socket. The TCP protocol is used to communicate with industrial equipment, composed of sensors, actuators (electro valves, lamps, relays), RFID readers, and manipulation robotic arms. The equipment connection is depicted in Figure 21.

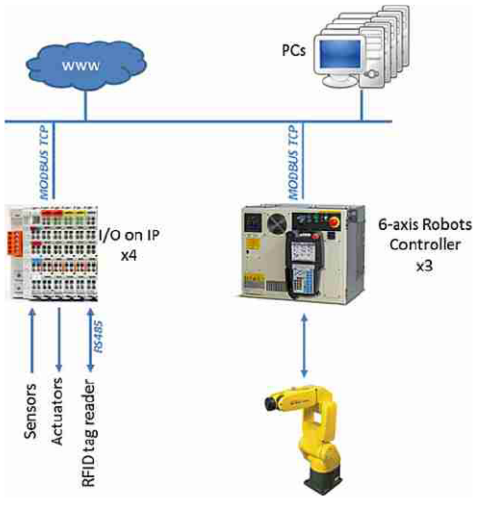


Figure 21 – SoHMS: SoHMS network, adopted from (QUINTANILLA *et al.*, 2016)

The virtual commissioning, the differential of this work, is performed via the parallel SoHMS integration with a DES tool, namely Rockwell ARENA. The manufacturing model in the ARENA software simulates the characteristics of operations and time demanded by the operations. This integration is advantageous to developers and stakeholders because it allows checking whether errors on the control system can be corrected before the actual implementation. The integration between the Rockwell ARENA and the SoHMS is depicted in the figure 22.

The authors conclude that a higher level of generalization in the architecture interfaces and a better definition of the holons' role is necessary.

The SoHMS and the virtual commissioning concept presented in that work are essential for defining this dissertation architecture interface. As presented previously, the virtual commissioning concept matches with the specific object of integrating a DES tool to the control architecture prototype. The SoHMS will not be used entirely, but it gives inspiration for the user and operator interfaces to access the holonic system functionalities. The model utilized before on the DES tool presents a static environment. This dissertation plans to escalate the model to configure a more dynamic behavior, where resources are defined as a composition of other resources, and the combination of resources provides variable capabilities.

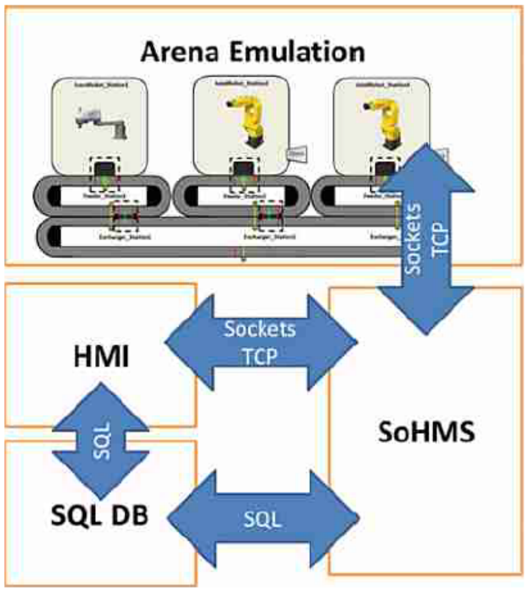


Figure 22 – SoHMS: Virutal-comissioning Integration, adopted from (QUINTANILLA *et al.*, 2016)

4.6 ORCA-FMS: a dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling

In (PACH *et al.*, 2014), the authors proposed the dynamic Architecture for an Optimized and Reactive Control (ORCA). ORCA hierarchical part is responsible for global optimization, while the heterarchical element allows local optimization. ORCA can handle predictive and reactive behaviors simultaneously.

The ORCA organization is composed of three layers: the physical system (PS), the local control (LC), and the global control (GC). While the global control has a global view of the system, the local control has only a local view, and its goal is to react to unexpected events occurring in the physical system. The local control has the autonomy to detect any perturbation and trigger the disrupted mode. Then, it controls the entity's responsible behavior and begins the optimization. The global control does not have access to this control because it occurs locally.

In the article, ORCA is applied to a Flexible Manufacturing System (FMS) called ORCA-FMS. Each machine belonging to the system can provide services to complete interactive operations. This process is explored, solved, and handled statically and dynamically by ORCA-FMS (PACH *et al.*, 2014).

The authors carry out an experimental study. The aim was two-folded: highlight the advantages of using hybrid control architectures and demonstrate the feasibility of ORCA-FMS. The experiment comprises two phases: a part produced under system perturbations; a part produced without system perturbations. The set of operations is assumed to be known at the beginning of the production.

To compare the final ORCA-FMS behavior, the authors executed a comparative study, whose results are shown in figure 23. The ILP* mode represents the system's performance when never disrupted (following an optimal global schedule). The potential fields model (PF model) shows the system performance when the system initialization applies the compensation strategy. The ILP model represents the system reacting just when a failure is introduced. The ORCA-FMS shows a significant improvement regarding the first completion time compared to the potential fields and ILP model.

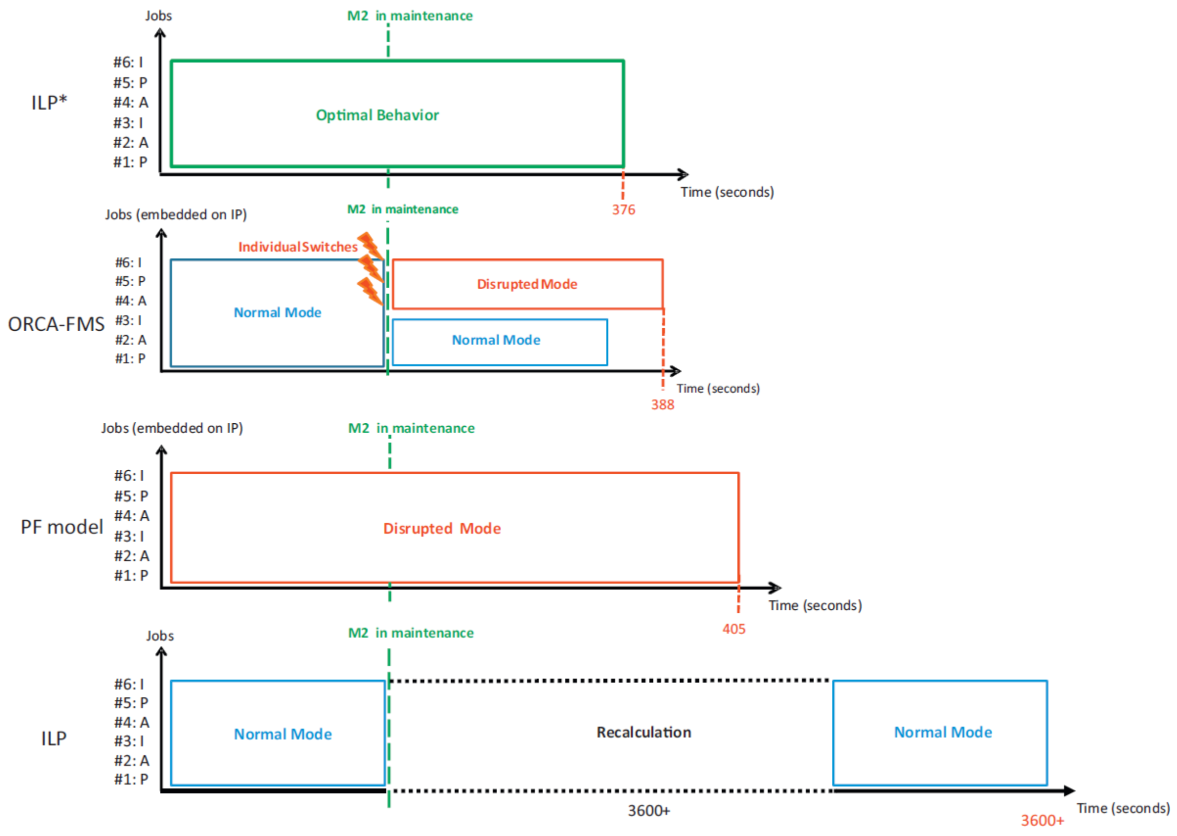


Figure 23 – ORCA behavioral comparison, adopted from (PACH *et al.*, 2014)

4.7 ARTI (Activity Resource Type Instance)

Title: **ARTI Reference Architecture – PROSA Revisited**.

In the ARTI architecture proposal (VALCKENAERS, 2018) the author extends the well-known and referenced PROSA architecture creating in the process the ARTI – Activity Resource Type Instance – architecture. The paper details the process of the PROSA conceptualization. The reason that first influenced the author to extend the PROSA architecture was the nomenclature. Even in the HMS domain, the common understanding of an order or product holons sometimes leads to misunderstanding. The

ARTI architecture supports the PROSA team and improves the PROSA architecture in other aspects, such as artificial intelligence (AI). Namely, the AI features included are:

- Non-optimal holon aggregation: The Nobel Prize winner Herbert Simon stated that flexible aggregation hierarchies are crucial/essential to adapt within time windows. Based on that, as in PROSA, the ARTI the aggregation is a non-optional feature;
- User mass consideration: the former architecture has not foreseen the user mass usage over the applications, which supports the whole product life cycle, helping in the improvement of the whole production chain. The ARTI considers the user mass and sets a limit to the architecture, where self-reinforcement learning should start;
- Intelligent beings: instead of using intelligent agents, the ARTI architecture designs the decision-making mechanism via intelligent beings. The mechanism that takes the decision is seen as a repository of useful tools, which intelligent beings can use to accomplish a specific objective;
- Short-term forecasts: the ARTI considers that when a decision is made, the outcome should be predicted and informed to the user who shares the use of a specific working scenario. That scenario can be considered as an imagination process. The imagination process is another feature included in the ARTI architecture.

The nomenclature problem was solved using a more generic terminology: order holons become activity holons; product holons become activity types holons; resources holons are subdivided into instances and types holons. The ARTI architecture organization is described in the figure 24.

The intelligent agents (green cubes) on the picture represent the technologies used to fulfill the intelligent being requirements, while the yellow cubes represent the connection of the intelligent agents to the rest of the system. The blue cubes of the architecture represent the system flexibility to the intelligent beings' descriptions (e.g., activity type, activity instance, activity instances grouped to represent a resource activity). Human beings are seen as activity performs, covering all the cubes on the architecture.

The ARTI addresses the in-depth interoperability of the capabilities of a Digital Twin. The ARTI separates digital and physical and supports using the proposed architecture to develop Digital Twins. The architecture contributes to the Digital Twin adoption with a generic nomenclature that should avoid misunderstanding and disconnect the Digital Twin from a specific field, extending the ARTI to other fields other than manufacturing.

The author concludes the paper by emphasizing that the content should be considered work-in-progress and open to new business ideas and suggestions. The author

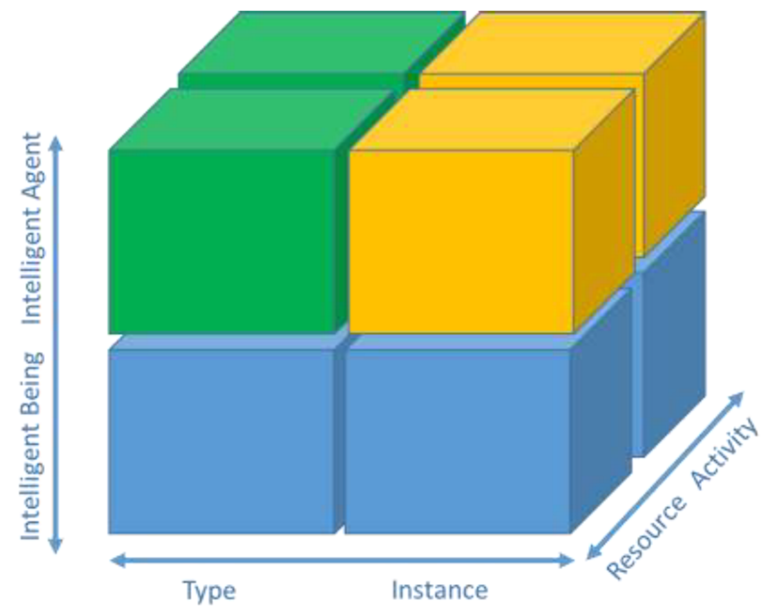


Figure 24 – ARTI: Architecture, adopted from (VALCKENAERS, 2018)

puts focus on the paper review process and how the research community could benefit whether the comfort zones (e.g., closed research communities) were more flexible and open to communication.

Though considered by the authors as a working-in-progress, the ARTI architecture promotes the idea of generalization replacing the names of the holons proposed on PROSA and proves to be a starting point to the more generic and applicable architecture for Digital Twins. The ARTI architecture gives a critical insight into the current development regarding the use of the platform as a Digital Twin, enabling competitive advantages and benefit from the data collected by a system following the ARTI architecture guidelines. As with PROSA, the ARTI is a very generic architecture, not defining complex interactions and the data exchanged by the holons but is an architecture that addresses the multi-level characteristics of the holons. The architecture proposed by this dissertation does not require a high level of abstraction and can follow the PROSA architecture as it was designed for the manufacturing domain.

4.8 Holonic architectures review

According to the analyzed HCAs, the following points were noticed:

- high-level holons definition such as the product, resource, order, and staff holons;
- high-level interaction model definitions between holons (e.g., order and product holons);
- specialized holons such as the disturbance, state, holonic manufacturing systems

holons;

- switching mechanism to drive the transition between stationary and transient states (e.g., disrupted state) in the holonic control architecture;
- use of a reasoning mechanism used to identify evolutionary system opportunities and reorganize the system to solve the fault adaptability problem of a manufacturing system;
- use of governance rules "what-if" to detect and adapt to fault scenarios (e.g., rush order, product defect, resource fault);
- association between SoA and HMS to interface an HMS with operators, customers, and other HMS;
- use of virtual commissioning to evaluate the HMS architecture using a DES tool previous to the actual configuration and operation;
- lack of a holonic architectures develop under an ontology specification which considers the separation between physical and cyber worlds;
- lack of the definition of interactions necessary to build up dynamic stations necessary to the LMAS paradigm;

Accordingly to the observed features, this dissertation aims to design a holonic architecture with the following additional features:

- definition of specialized holons as sub-classes of the PROSA definitions (e.g., scheduler holon as a subclass of a staff holon);
- use of the switching mechanism idea to identify a failure and to manage the system during a failure (e.g., fault in station assembly equipment);
- application of a data model derived from an ontology that considers HMS, factory, resources, and products concepts;
- definition of the interactions necessary to build up dynamic stations required by the LMAS paradigm.

A qualitative classification of how the studied literature attends the LMAS requirements (table 2) is presented in the table 3.

Table 3 – HCA literature review.

	ID	**Requirements**	**PROSA**	**ADACOR**	**ADACOR^2**	**Pollux**	**Quintanilla et al.**	**ARTI**	**ORCA**
Resource	RQ-01	Multipurpose station and resources	◕	◑	◑	◑	◔	◕	◑
	RQ-02	Fabric model / Shop floor management	◕	◕	◕	◑	◔	◕	●
	RQ-03	Mobile resources / temporary station locations	◑	◔	◔	◔	◔	◑	◑
	RQ-04	Spatio-temporal order relations between stations and resources	○	○	○	○	○	○	○
Product	RQ-05	Products, parts and assemblies structures (physical relationships)	◑	◑	◑	◔	◕	◑	●
	RQ-06	Products, parts and assemblies manufacturing data and information	◕	◕	◕	◔	◕	◕	●
Process	RQ-07	Individual order routes	◔	◔	◔	◔	○	◔	◔
	RQ-08	Interface to transportation services	◔	○	○	○	○	◔	◔
Design patterns	RQ-09	Global manufacturing plan	◑	◕	◑	◕	◔	◑	●
	RQ-10	Strategies for delay and failure compensation	◔	◕	●	●	○	◔	◕
	RQ-11	Central control / Central monitoring	◑	◕	◑	◕	◔	◑	◕
	RQ-12	Decentralized order execution	●	●	●	◑	◑	●	●
	RQ-13	Ontology-based	○	◔	◔	○	○	◔	○
	RQ-14	Type/Instance	◔	○	○	○	○	●	○

4.9 The ADACOR ontology

Title: **The Role of Foundational Ontologies in Manufacturing Domain Applications**.

In this work, the authors investigate the use of the foundation ontologies on top of more specific ontologies in real applications and propose an extension to a selected foundation ontology to suit a previously described HMS architecture, ADACOR. The work's objective is to show how foundation ontologies, the most general ones, are essential for system reusability and how they can be extended to more specific ontologies (BORGO; LEITÃO, 2004).

The authors performed a state of art study review on foundation ontologies like DOLCE (the Descriptive Ontology for Linguistic and Cognitive Engineering), OCHRE (the Object-Centered High-level Reference Ontology), OepnCyc, SUMO (the SUggested Upper Merged Ontology), BFO (the Basic Format Ontology). As criteria for selecting the foundation ontology, the authors considered the richness of conceptual distinctions, at least a few related to the aimed domain (manufacturing), and the characterizable aspect of the features declared. Based on that criteria, the authors selected the DOLCE ontology (BORGO; LEITÃO, 2004).

The authors proposed the ADACOR ontology based on the FIPA ontology service recommendations. The ontology is described in an object-oriented, frame-based manner, representing concepts and predicates. Apart on the ADACOR ontology is depicted in figure 25.

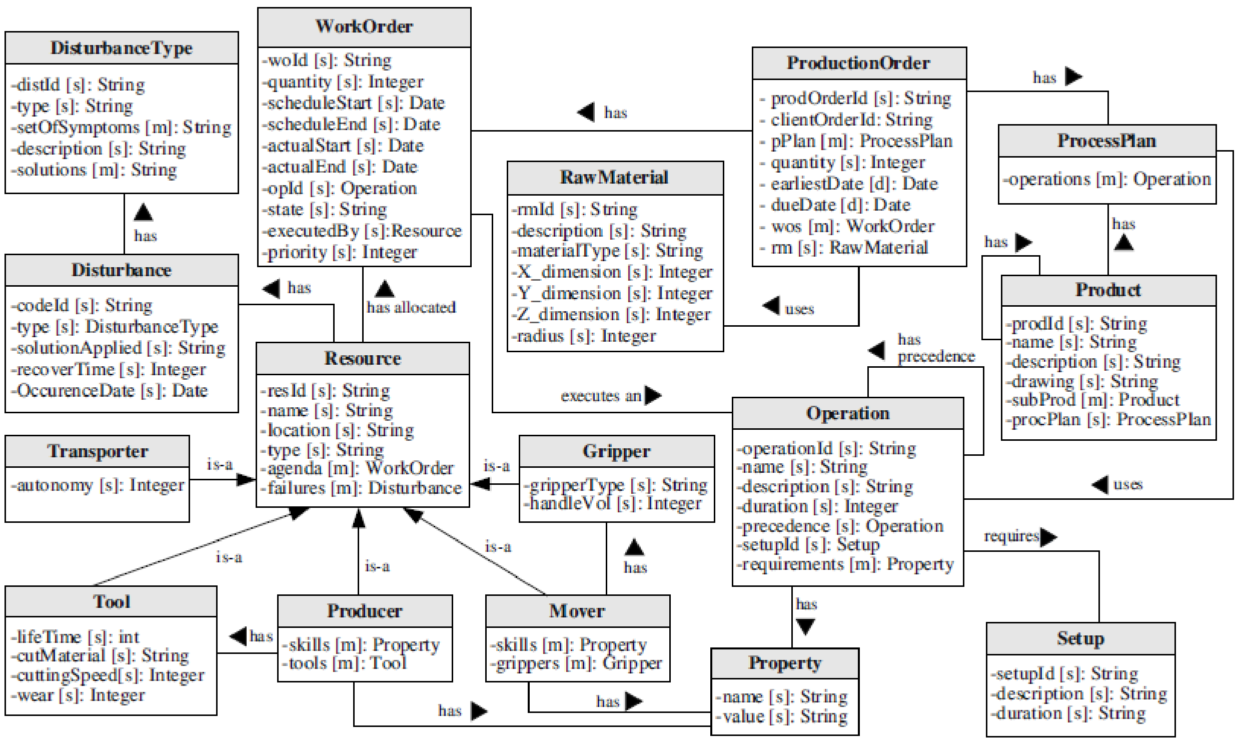


Figure 25 – ADACOR ontology, adopted from (BORGO; LEITÃO, 2004)

The authors provide a detailed study of relations between the classes and predicates of DOLCE and ADACOR ontologies. They found, in this manner, a high degree of similarity between the architecture and the foundation ontology.

The ADACOR ontology brings important attribute definitions and helps in the high-level phase of the proposed ontology conceptualization. It provides just a section of the ontology given in a class diagram not explicitly defining the holons' properties and relations. The current work aims to extend the ontology resource definition to a more flexible approach where the resource is a component of a station holon, where the assembly operations are executable.

4.10 Unified ontology for a holonic manufacturing system

Title: **Unified ontology for a holonic manufacturing system**.

The authors provide an extensive literature review on different proposals to HMS architectures and ontologies used in the manufacturing context in this research work. Reviewing and analyzing the former HMS works, the authors proposed one unified ontology that integrates roles and behaviors. The ontology validation was held in a case study in a manufacturing cell (SIMÓN-MARMOLEJO; RAMOS-VELASCO, 2018).

The state of art research analyzed various holonic manufacturing architectures: PROSA, ADACOR; MES-MHS, ADACOR^2, MAS-DUO & RFID-IMS and H2CM (Holonic Hybrid Control Model). As a result, except for ADACOR, the authors indicate that the observed architectures do not propose a formal manufacturing ontology.

Detecting the need for ontology features integration into holonic HMS, the authors selected seven different research works on the manufacturing field's ontology area to analyze. The analysis was based on specific criteria: manufacturing field converging, concepts formalization, normalized and semantic data, hierarchical levels, evaluation of proposed ontology, evidence of the application on HMS/MAS systems, taxonomic relationships, holon or agent definitions, attributes, and predicates.

These criteria form the group of requirements the authors proposed to integrate with their architecture as, according to their analysis, none of the seven analyzed architectures provides a unique solution. The figure 26 represents the high-level class of their proposed ontology.

In the current dissertation, the Unified Unified ontology for a holonic manufacturing system will be extensively used as it presents holonic definitions such as disturbance, holonic manufacturing system, state and administration, holons. These types of holons do not exist in previously analyzed architectures and ontologies. The holons are defined at a high level in previous works, as in PROSA. The Unified ontology also defines suggestions for naming the attributes which compose the holons. The current dissertation will explore the holonic definition presented in (SIMÓN-MARMOLEJO; RAMOS-VELASCO, 2018) and extend to an ontology where the explicit separation between the cyber and physical parts is present.

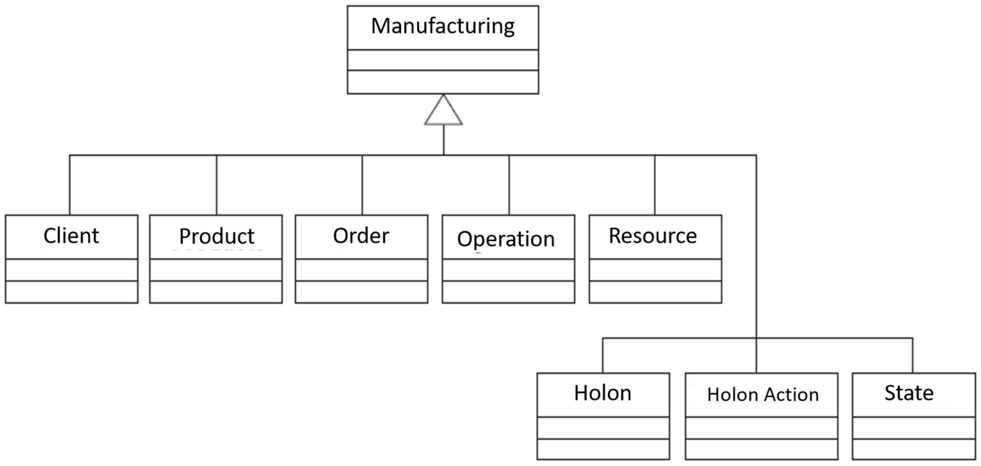


Figure 26 – First and second levels of unified ontology proposal, adapted from (SIMÓN-MARMOLEJO; RAMOS-VELASCO, 2018)

4.11 MaRCO Ontology

Title: **The development of an ontology for describing the capabilities of manufacturing resources**

In (JÄRVENPÄÄ *et al.*, 2019) the authors propose a manufacturing ontology called Manufacturing Resource Capability Ontology (MaRCO). The MaRCO ontology was built following a product model, process model, capability model, and resource model (MaRCO). Figure 27 represents the MaRCO composition.

The MaRCO ontology is focused on the resource level and the definition of resource organization and capabilities. The ontology allows its users to represent dynamic systems formed by devices that can be a group of devices that may offer additional capabilities based on the component devices' capabilities. The MaRCO ontology considers the dynamic capability changing depending on how a device is composed. Figure 28 represents that a Station in a manufacturing Site context may be composed of Device Combinations and Individual devices. The individual devices have capabilities that represent a combined capability when the devices are merged.

The MaRCO ontology was created utilizing the OWL language and tested utilizing the Protègè software through the application of SPARQL queries.

The MaRCO ontology will be used as a basis for developing and applying the so-called LMAS Ontology. The MaRCO ontology will represent the concepts of the physical world as it carries concepts of factory organization and dynamic device configurations required by the LMAS formulation. The MaRCO ontology will be extended to grasp holonic concepts of the previous ontologies that were not covered on the MaRCO ontology as it was not the authors' original intent.

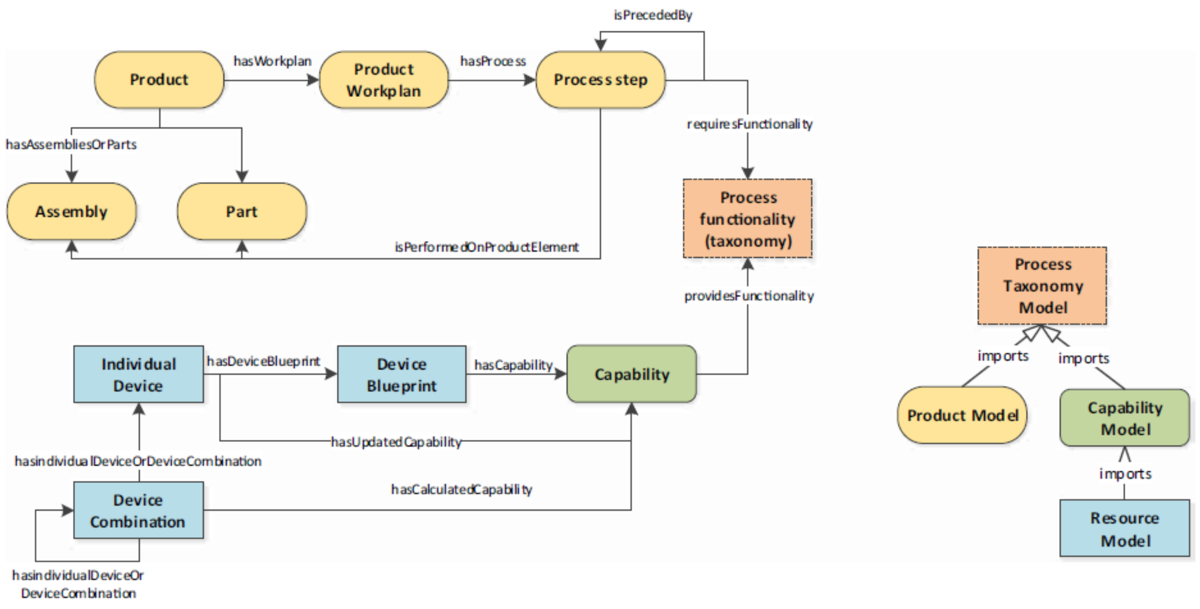


Figure 27 – MaRCO ontology composition, adopted from (JÄRVENPÄÄ *et al.*, 2019)

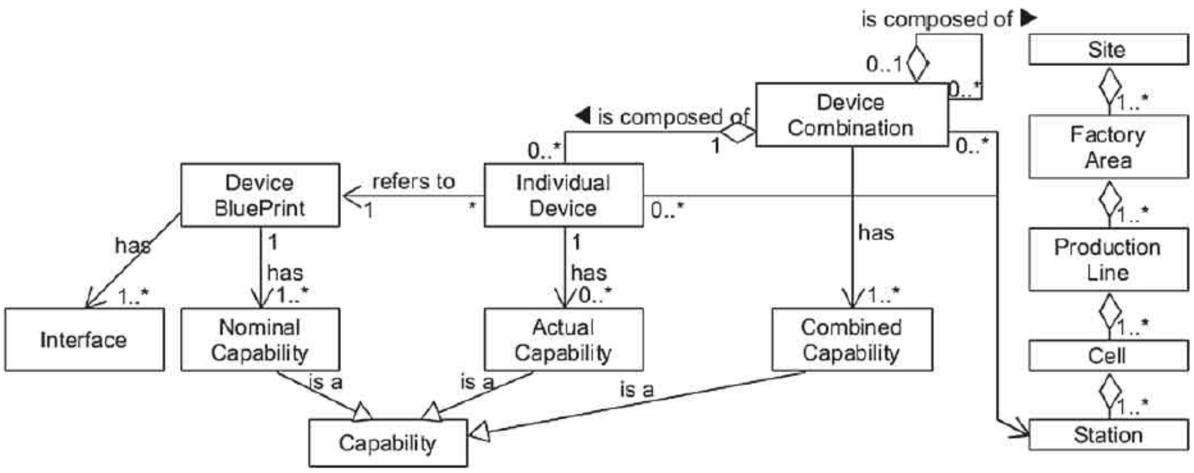


Figure 28 – Section of the MaRCO ontology, adopted from (JÄRVENPÄÄ *et al.*, 2019)

4.12 Ontologies review

Regarding the manufacturing ontologies observed, what could be noticed were:

- conceptualization related to high-level holons definition such as the product, resource, order, and staff holons;
- concepts, relations, and attributes related to specialized holons such as the scheduler, disturbance, state, and holonic manufacturing systems holons:
- attributes suggestions to general classes of holons such as resource, scheduler, disturbance, and operation holons;
- significant amount of capabilities definition;
- concepts, relations, and attributes related to the dynamic behavior of devices formed upon by other devices;
- concepts, relations, and attributes related to factory organization (e.g., station, line, site);
- lack of availability to the research community of an ontology file (e.g., OWL) for HMS;
- not consideration of an ontology that separates the cyber and physical world, while considering HMS concepts.

Based on the researched works on ontologies, during the data model development, this dissertation will contribute to:

- develop an ontology implementation with HMS concepts to grasp the LMAS requirements;
- through a merge between an HMS ontology and a resource-oriented ontology, create an ontology which explicitly separates the cyber and physical worlds;
- explicit instantiate a reduced data model from the developed ontology.

The LMAS requirements (table 2) inspire the necessary knowledge that should be presented in a suitable ontology. The used ontology should contain concepts related to: individual order holons, temporary stations, mobile resources, transport, and organization levels (e.g., shop floor, station, line). A qualitative classification of how the studied ontologies attend the LMAS requirements is presented in the table 4.

Table 4 – Ontology literature review.

Ontologies	**Focus**	**Language**	**Individual order routes**	**Temporary stations**	**Mobile resources**	**Transport**	**Organizations levels**
ADACOR Ontology	Interoperability with foundational ontologies	-	◕	◔	●	●	○
Marmolejo et al.	Reuse of ontologies for HMS	-	◔	○	●	◕	◑
MaRCO	Resource and reconfigurability	OWL	◔	◕	◕	◕	●
MASON	Concepts of industrial manufacturing	OWL	○	○	○	○	◕
MSDL	Manufacturing Services	OWL	○	○	◕	◕	◕
BaSyS	Capability representation	OWL	○	○	◑	◕	○

5 System architecture design

In this chapter, the ANEMONA analysis phase results are presented. The analysis phase produces requirements, simplifications & assumptions, system goals, and architecture models. The architecture models are composed of the organization, interaction, and agent models. The agent model is complemented with the agent state model, which is not a requirement in the ANEMONA methodology but valuable to represent the agent behavior from an external and internal point of view. The reasoning for choosing the ANEMONA methodology, details about its development cycle, and the detailed definition of its output architecture models are presented in section 3.4.

5.1 Architecture analysis

The architecture analysis phase consists of identifying the system requirements, assumptions and simplifications, and goals.

5.1.1 Requirements

The architecture requirement specification is based on the LMAS requirements presented in section 3.7.3 and respects the use case scenarios presented in section section 3.7.2. Requirements and use cases provide an essential idea about the necessary interactions, functions, and variety of holons. Some of the main requirements of table 3.7.3 are detailed below:

- R6 and R7: these requirements are related to the station setup process. It is considered that the stations should execute a setup step before being able to cooperate with order holons executing operations;
- R11: when a station finalizes all the operations on its schedule, it should dissolve the assembly equipment used for its setup process. The assembly equipment can be later used for subsequent configurations;
- R15: the station holons contain a recipe that describes the operation types that the station can execute. This differentiation among the station holons challenge the scheduler holon, which needs to optimize and find a suitable plan respecting all customer order input and the station capabilities;
- R22: the operation should provide the configuration files to the broker holon during initialization. The configuration files describe how the shop floor, station, and assemblies are described in terms of physical characteristics and capabilities;
- R34: the scheduler holon needs to interpret the data given by the control layer. From the perspective of the scheduler holon, the control layer takes the form of

the broker holon. The broker holon should understand and correctly interpret the data given in by the operator;

- R36, R39, and R40: these requirements are associated with the station holon's ability to react locally to a failure imposed by one of the assembly equipment composing the station setup. The station should be able to replace the defective assembly equipment;
- R43: the broker holon, as the main actor of the control layer, should be responsible for all the essential and supportive holons initialization;
- R49: The order holon should create a sub-assembly holon at the beginning of every job execution (set of operations). The sub-assembly holon represents the assembly during its manufacturing process.

Table 5 – Detailed architecture requirements.

ID	Related	Requirement
R1	RQ-01	The broker holon should provide a service for the operator add assembly equipment to the system
R2	RQ-01	Assembly equipment holons should provide build station service
R3	RQ-02	Station holons should register the real starting and ending time of operations
R4	RQ-02	The station holons should subscribe to the shop floor holon when initializing
R5	RQ-02	The assembly equipment holons should subscribe to the shop floor holon when initializing
R6	RQ-03	Station holons should provide a autonomous mechanism to execute the station setup once the optimal schedule is received
R7	RQ-03	Station holons should complete their setup process before being able to execute operations
R8	RQ-03	Station holons should start their setup process when the simulation time is started
R9	RQ-03	Station holons are considered setup once the assembly equipment are in place and the setup time counted

R10	RQ-03	Station holons should consider a setup time to simulate the assembly equipment's adjustments process
R11	RQ-03	Station holons should provide a service to dissolve the assembly equipment from its station configuration
R12	RQ-03	Station holons, when in IDLE state, should be able to execute assembly operations
R13	RQ-03	Station holons should automatically disolve, when all the operations in its schedule were executed
R14	RQ-03	Station holons should register the real starting and ending time of the station setup process
R15	RQ-03	Station holons can only executes operation type defined in their station type
R16	RQ-04	Station holons schedule should at minimum contain: operations planned starting and ending times; assembly equipment used for station setup
R17	RQ-04	Station holons should provide a status update server to the order holons
R18	RQ-05	The part type holon should include physical and dynamic features of parts
R19	RQ-05	The assembly type holon should include physical, dynamic features, necessary parts and precedence graphs
R20	RQ-06	The capability holon should provide to their clients the performance information delivered by each station type including operation type and processing time
R21	RQ-06	The capability holon should work cooperatively with the station type holon to inform its clients about the station capabilities
R22	RQ-06	The operator should provide at the system initialization the path of configuration files including: shop floor description, station types, part types, assembly types, assembly equipment type and assembly equipment
R23	RQ-07	Assembly equipment holons should provide provide transportation services

R24	RQ-08	Assembly equipment should provide feedback about the current motion progress while the setup process is executed
R25	RQ-08	Assembly equipment should provide feedback about the current motion progress while in motion
R26	RQ-09	Station holons configuration should be provided as part of a global manufacturing plan
R27	RQ-09	The user interface holon should provide means to the operator specify a customer input order and optimization execution flags: generate completion graph, execute animation, calculate BIG_M, calculate C_MAKE and priority.
R28	RQ-09	The user interface holon should parse the incoming information and register to the database holon
R29	RQ-09	The broker holon should parse the shop floor, operation, station and assembly data, optimization flags and request the optimal schedule to the scheduler holon
R30	RQ-09	The scheduler holon should save the optimization decision variables data to a file the the local optimization history folder
R31	RQ-09	The scheduler holon should register the raw and the internal parsed data to a local optimization history folder
R32	RQ-09	The scheduler holon should generate the animation and completion diagrams when required and save them to the local optimization history folder
R33	RQ-09	The scheduler holon should provide feedback about the optimization progess.
R34	RQ-09	The scheduler holon should parse the optimization data and send back the optimal schedule to the broker holon
R35	RQ-10	Station holons should provide pause, unpause, resetup and replan services
R36	RQ-10	Station holons should provide a mechanism to replace a defective assembly equipment

R37	RQ-10	Station holons should provide a service to update schedule
R38	RQ-10	Station holons should assume a FAILURE state when a assembly equipment fails
R39	RQ-10	Station holons should provide the replace mechanism just for assembly equipment already the setup area
R40	RQ-10	The station holon should, while in a replacement operation, call a assembly equipment from the same type as the defective
R41	RQ-11	The broker holon should offer services for the operator to start, pause, reset and set the simulation time
R42	RQ-11	The shop floor holon should provide the current state of the subscribed station and assembly equipment holons
R43	RQ-11	The broker holon should intialize the database, station type, part type, assembly type, assembly equipment, type, capability, shop floor, switch mechanism, user interface, holonic manufacturing system, part, assembly, assembly equipment, station and order holons
R44	RQ-11	The broker holon should distribute the optimal schedule to the assembly equipment, part, station and order holons
R45	RQ-11	The broker holon should send the unused assembly equipment to a resource area
R46	RQ-11	The broker holon should provide mechanisms to load optimization results
R47	RQ-11	The broker holon should provide services to reset holons: delete all operation holons (station, order, part, and assembly holons), delete the operation holons parameters
R48	RQ-11	Holonic manufacturing system holon should provide a system snapshot service, containing the current state and data properties of all the subscribed holons
R49	RQ-12	Order holons should create a sub-assembly holon when the first operation is executed
R50	RQ-12	Part holons should subscribe to their related assembly and order holons

R51	RQ-12	Assembly holons should subscribe the their related order holons
R52	RQ-12	The user interface holon should send the parsed data to the broker holon
R53	RQ-12	The order holons should execute instances of a general operation class
R54	RQ-12	The station holons should execute instances of a general operation class
R55	RQ-13	The holon data class data attributes should follow the ROS message definition based in Holontology
R56	RQ-13	The data attributes definition should be based in a higher level ontology containing concepts pertinent to the cyber and physical worlds
R57	RQ-14	Station holons should assume only one station type
R58	RQ-14	The station, assembly equipment, part, assembly holons instances should contain the generic definition of a respective type
R59	RQ-14	The station type definition should define a blue print for real instances depending on assembly equipment types availability
R60	RQ-14	Operations should be instances of a operation class including the operation type
R61	RQ-14	Capability, part type, assembly type, assembly equipment type, station type holons should hold the system knowledge
R62	RQ-14	The assembly equipment type holon should include the assembly equipment type

5.1.2 Simplifications and assumptions

The table 6 presents the simplifications and assumptions categorized in LMAS Planning or Planning & Control. Some of most relevant simplifications and assumptions are:

- SA01: the assembly priority graph is known to the broker holon, so the broker holon uses a topological sort algorithm to create one feasible linear sequence per product and passes that information to the scheduler holon;
- SA03: if the station holon requires an equipment replacement, only equipment of the same type can be used for replacement. Besides being of the same type, the equipment should also belong to the group of spare equipment;

- SA04: for simulation purposes, the part holons are considered already available to the station. This simplification easy the fleet management simulation that does not need to carry the operations for parts transportation;
- SA09 and SA10: even though the station has a 2D internal grid for allocation of resources, during the simulation, the problem that concerns the allocation of these resources in this internal grid is not provided by the architecture models;
- SA11: one station can execute only one operation at a time. The parallel execution is just possible when two or more stations are considered. Even though, for one assembly, there is no possibility to have operations executed in parallel since the scheduler holon follows a linear precedence order to derive the optimal schedule;
- SA13: the human factor is excluded from the models due to the assumption of fully automated and autonomous LMAS;
- SA14: all the functionalities should be provided to the models through a ROS interface;
- SA16: all the operations concerning an assembly are just dependent on the operation for this same assembly. There is no dependency between assemblies.
- SA19: this means that only failures in individual assembly equipment are handled instead of general failures or failures derived from a higher-level analysis (e.g., an average of the difference between scheduled and executed operations)
- SA23: the station starts the station setup process (call for the assembly equipment to setup) as soon as the time simulation starts;

Table 6 – Simplifications and assumptions

ID	Category	Name	Description
SA01	Planning Control	Assembly priority graph	The priority graph for an assembly is known. The priority is graph is input for a topological sort algorithm that derives a linear sequence.
SA02	Control	Transporters	Only assembly equipment of type “Transporter” are allowed to transport material between stations. Transporters are finite and do not build stations.

SA03	Control	Spare types	In case of system disruption, when it is necessary to replace a assembly equipment in a station, only types of Type + _SPARE are suitable for the replacement.
SA04	Control	Part holon location	Part holons are considered to be located in the working stations. There is no transport resolution for parts transportation.
SA05	Control	Sub assembly holon	Each assembly holon is associated with only one sub assembly holon, and the assembly holon should ask transportation services to move between stations.
SA06	Planning & Control	Transport times	The distance between locations is given using the Manhattan distances. The distance map is provided globally.
SA07	Planning & Control	Transport capacity	Only one sub-assembly is allowed to be transported in a Transporter.
SA08	Planning & Control	Shop floor abstraction	The shop floor is discretized using an equal area squared approach (results in a 2D grid).
SA09	Planning & Control	Station occupancy	A station occupies a shop floor location. The station intra-organization (station configuration) is divided into a 2D grid design pattern. The internal 2D design pattern is not considered in the interactions.
SA10	Control	Location-to-location transport	The time taken to transport a sub-assembly is considered only from the high-level shop floor abstraction, the station internal configuration is not considered.
SA11	Planning & Control	Non-parallel station execution	A station can only execute one operation at the same time.

SA12	Planning & Control	Necessary conditions for operation	To execute an operation, it is necessary that the station is setup, parts are in place, sub assembly is in place.
SA13	Planning & Control	Human factor	The human factor is excluded from the architecture considerations.
SA14	Control	Agent connectivity	All agents are considered to have an interface based on the ROS framework.
SA15	Control	Safety and security	Safety or security are not considered in this work.
SA16	Planning & Control	No assembly mutual dependency	There is no operation dependency between different assemblies.
SA17	Control	Mobile assembly equipment continuous operations	Battery consumption is not considered. The mobile assembly equipment is considered to be available 100 % when not in FAILURE state.
SA18	Planning & Control	System units	spatial: m; time: s; mass: kg. All derived units should respect the above fundamental units.
SA19	Control	Disruptions	Only individual assembly equipment have handling of disruptions.
SA20	Control	Disruption handling	Only local assembly equipment replacement and total system reconfiguration are considered as strategies for system disruption compensation.
SA21	Control	Disruption action	Disruptions cannot be simulated while an operation is executed. In case a disruption is simulated while an operation is executed, the system waits for the operation to finish.
SA22	Control	Simulation time frequency	The simulation time advances at a frequency of 1 Hz.

SA23	Control	Station setup initial time	The station holons start their station setup process as soon as the simulation is started.
SA24	Control	Simulation time control	The system simulation starts when the simulation time starts to progress.
SA25	Control	Pick and place operations	For simulation purposes the pick and place operations at the beginning or ending of a transportation are not considered

5.1.3 System goals

The system goals are presented in the table 7. The system goals are derived considering the use cases and requirements. Some of the relevant goals are highlighted:

- O02 and O03: the broker holon here should provide a mechanism to interpret the system state (e.g., current assembly equipment locations), merge with the customer order input, and send the data to the scheduler holon that will derive an optimal global schedule
- O06 and O07: the systems should detect failures on assembly equipment and handle these disruptions with an appropriate strategy. The strategies are discussed in detail later in this chapter;
- O10: to test the HCA functionalities, a simulation should be available to the operator. In the simulation, the transport times and the operation times are simulated utilizing basic time functions in the chosen programming language (discussed later in the chapter 6).

Table 7 – Architecture system goals

ID	System Goal
O01	Execute the assembly process of large products in lot size 1 following the LMAS paradigm.
O02	Interpret customer orders and execute the optimization algorithm, merging the customer order requirements with the system status quo to derive a suitable assembly process schedule.
O03	Plan, throughout automatic optimization, the system assembly processes to define which operations for which orders are to be executed at a specific sequence, station, times, and location.
O04	Interpret the provided optimal schedule and launch the necessary autonomous manufacturing holons to drive the assembly process. The optimal schedule's operations and required station setups should be assigned to the launched holons in that process.
O05	Control the execution of assembly operations at the station level.
O06	Detect assembly process disruptions: assembly equipment failures, station failure and, lack of assembly parts.
O07	Handle the manufacturing system disruptions deciding whether a local compensation or a total system reconfiguration is required. A suitable heuristic should be used.
O08	Launch and control the system dynamic station setup process utilizing the assembly equipment available on the shop floor.
O09	Control and monitor the parts, sub-assemblies, and assembly transportation on the shop floor.
O10	Simulate the system's physical equipment behavior in order to test the holonic manufacturing system.
O11	Monitor the system assembly process execution

5.2 System overview

Considering the LMASs' dynamic re-configuration requirements (Section 3.7.3), figure 29 exhibits a layered, hybrid hierarchical-heterarchical approach (heterarchical systems adapt promptly to system disruptions due to their entities' autonomy, while hierarchical systems can be readily optimized because the system state is wholly known to a set of decision-making entities). The actor utilizes a user interface to communicate new orders to the broker holon. Orders are merged with current system knowledge, including description files and lower-level holon states. The result is sent to the planning holon. It comprises an LMAS-specific operations optimization model (c.f. (BUCKHORST; SCHMITT, 2020)). The broker holon interprets the resulting global schedules to derive individual schedules for execution layer holons: assembly, assembly equipment, station, shopfloor, order, and part holons. The operational holons cooperate when executing their local schedules following their interaction models: The overall system behavior results from cause and effect. The models represented at the right part of figure 29 establish the conceptual base for the interactions and roles exerted by the system holons. The organization model describes the holons' the dynamic structural organization; the environment model describes the external actions that trigger actions within the system architecture; the agent model describes for each holon its functions, state data, and goals; the tasks and goals model represents what functions are necessary to be executed to fulfill the system goals (c.f. table 7); and the interaction model shows the all the possible interactions between the system holons and their execution order

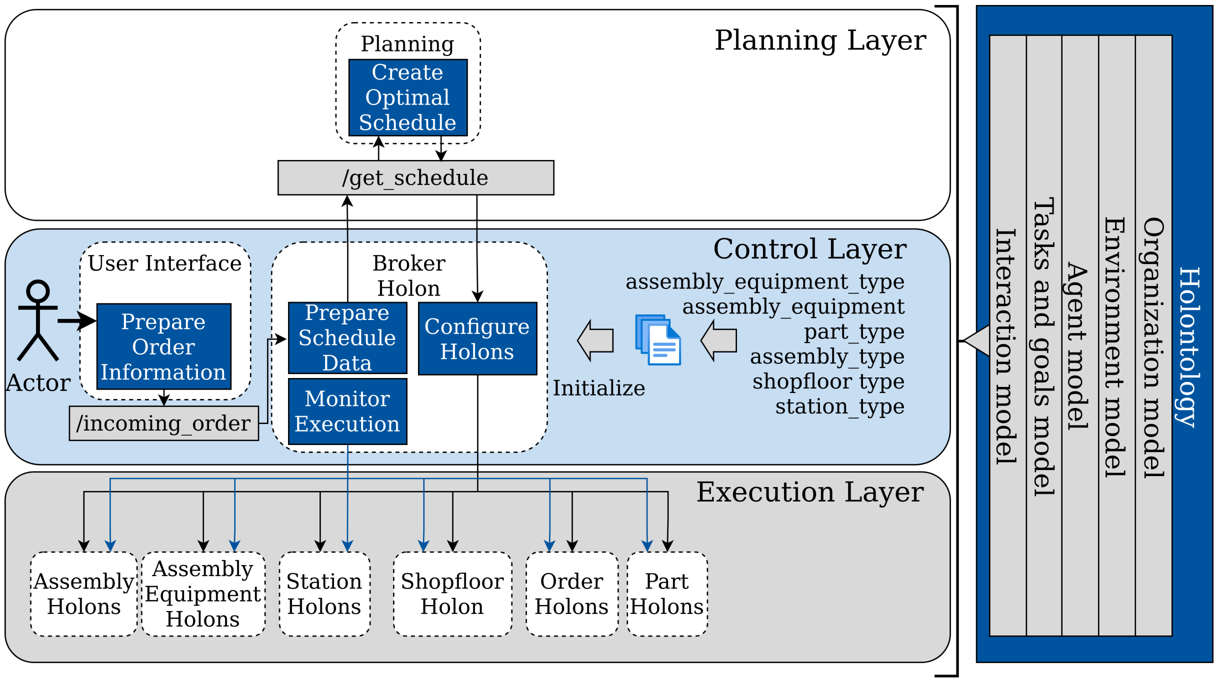


Figure 29 – LMAS HCA: Overview

5.2.1 Organization model

In figure 30 the ANEMONA organization model of the architecture, containing all high-level holons, is depicted. The holon names and functionalities are inspired by (BRUSSEL *et al.*, 1998; LEITÃO; RESTIVO, 2006; BORANGIU *et al.*, 2015; BARBOSA *et al.*, 2015). The main holons are summarized as follows:

- Scheduler holon: this holon interprets the current system state and generates optimal stations and assembly equipment schedules;
- Broker holon: this holon creates and deletes holons, receives customer orders, and monitors the assembly execution;

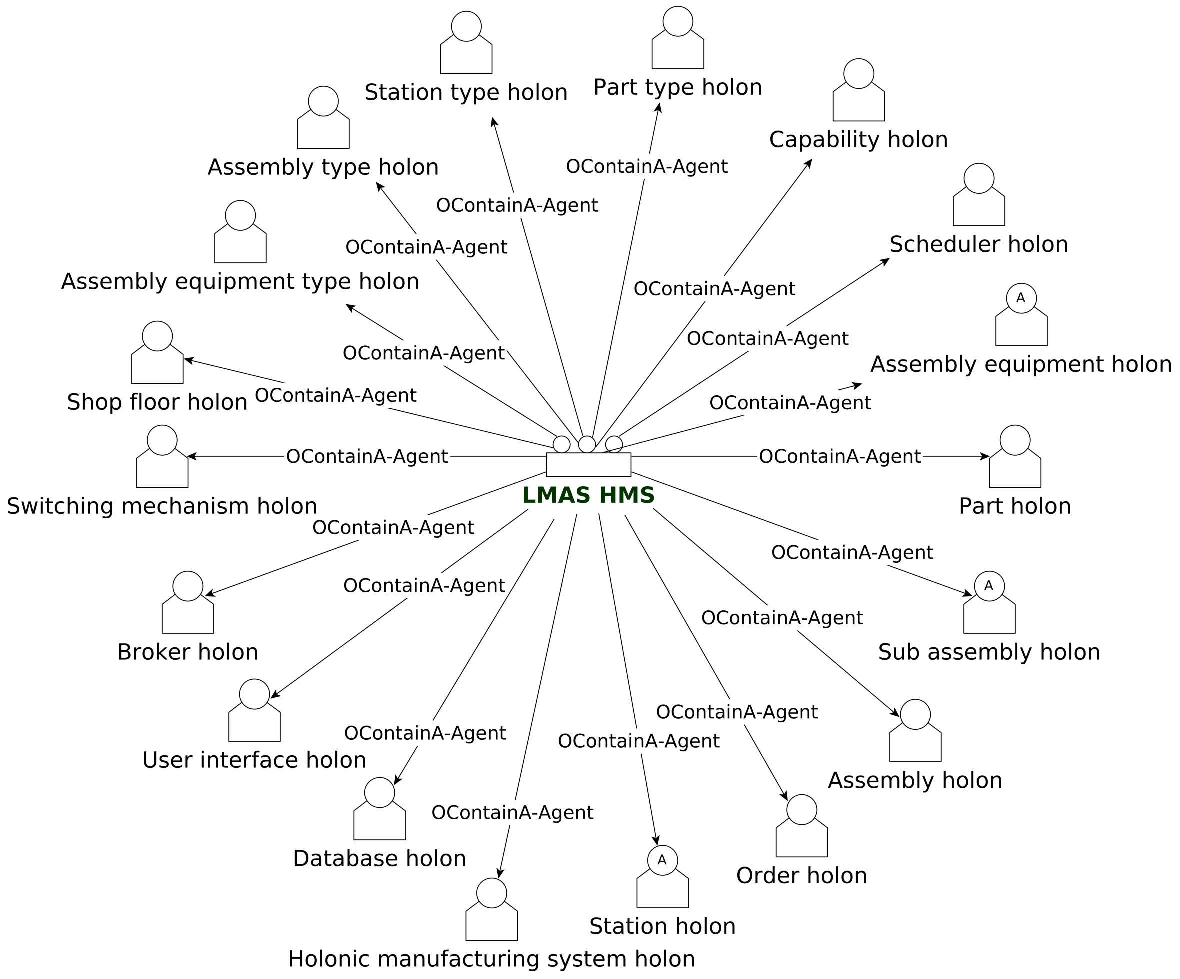


Figure 30 – HCA Organization Model (in ANEMONA Notation)

- Order holon (OH): this holon controls and monitors the execution of stations' operations;
- Assembly equipment holon (AEH): this holon represents equipment supporting assembly production operations (e.g., AGVs, tools, robots, fixtures, tool magazines);
- Station holon (SH): this holon represents a group of assembly equipment able to execute assembly operations. Each station holon has capabilities representing the offered performance to execute a set of operations;
- Part holon (PH): it represents a required piece necessary to perform an assembly operation;
- Sub-assembly holon(SAH): this holon represents a set of parts built together that are not a complete assembly yet;
- Assembly holon (AH): this holon contains the assembly "recipe" describing: operations necessary to be executed, how to connect the different parts spatially, and the list of parts that consists of the final assembly;

- Shop floor holon: it represents the shop floor properties and non-autonomous entities;
- Capability holon: this holon holds the station configurations' capability performances. A capability performance represents how the station configuration addresses a particular operation's execution (e.g., processing time);
- Type holons: these holons hold the type information of different entities on the shop floor, they are four: assembly type; resource type; part type; and station type;
- User interface holon this holon acts as an interface between the operations and the broker holon. Using this holon, the operator can input customer orders and provide configuration files to the architecture;
- Database holon: the database hoon offers generic write and reads services aiming to hold the system knowledge during the assembly execution;
- Holonic manufacturing system holon: this holon keeps track of all initialized holons and provides a snapshot service containing their current state. The scheduler holon uses the snapshot service before any optimization request;
- Switch mechanism holon (SMH): this holon provides mechanisms to alter the holon's behavior depending on incoming disruptions (E.g., rush orders, station breakdown, assembly equipment failure).

5.2.2 Agent model

Only the agent models of order (figure 31), station (figure 32), assembly equipment (figure 33), sub-assembly (figure 34), and part (figure 35) holons are presented for simplicity and noteworthy reasons. For each holon, an ANEMONA agent diagram is shown. The diagrams follow the ANEMONA notation representing the agents' functions, goals, and beliefs. Additionally, the agent state model describing the different states an agent may assume depending on the system dynamics is presented.

5.2.3 Agent model diagrams

The agent model diagrams are presented in the current section. The agent model diagrams capture each agent's functions, goals, and knowledge. The agent models are presented for order, station, assembly equipment, sub-assembly, and part holons ANEMONA agent model.

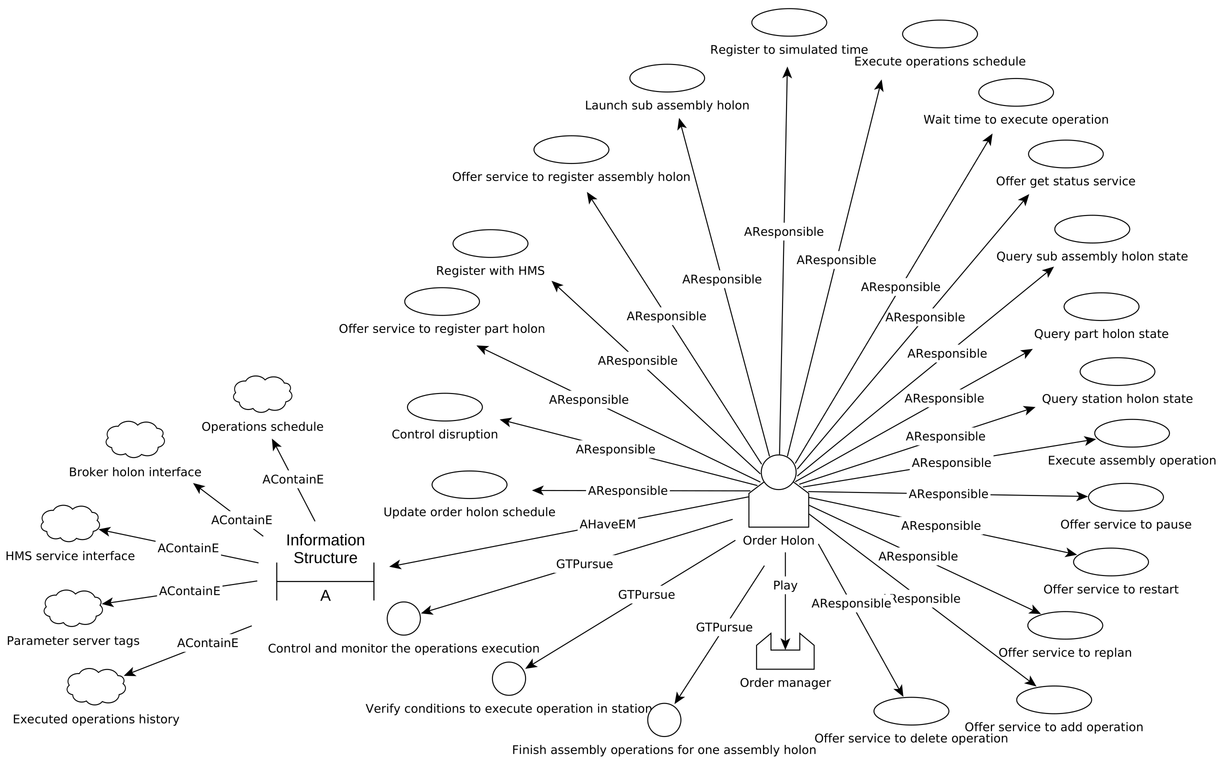

Register to simulated time
Execute operations schedule
Launch sub assembly holon
Wait time to execute operation
Offer service to register assembly holon
Offer get status service
Register with HMS
Query sub assembly holon state
Offer service to register part holon
Query part holon state
Operations schedule
Control disruption
Query station holon state
Broker holon interface
Execute assembly operation
Update order holon schedule
Offer service to pause
HMS service interface
Information Structure
A
Order Holon
Offer service to restart
Parameter server tags
Control and monitor the operations execution
Play
Offer service to replan
Executed operations history
Order manager
Verify conditions to execute operation in station
Offer service to add operation
Finish assembly operations for one assembly holon
Offer service to delete operation
AResponsible
AContainE
AHaveEM
GTPursue

Figure 31 – Order holon agent diagram.

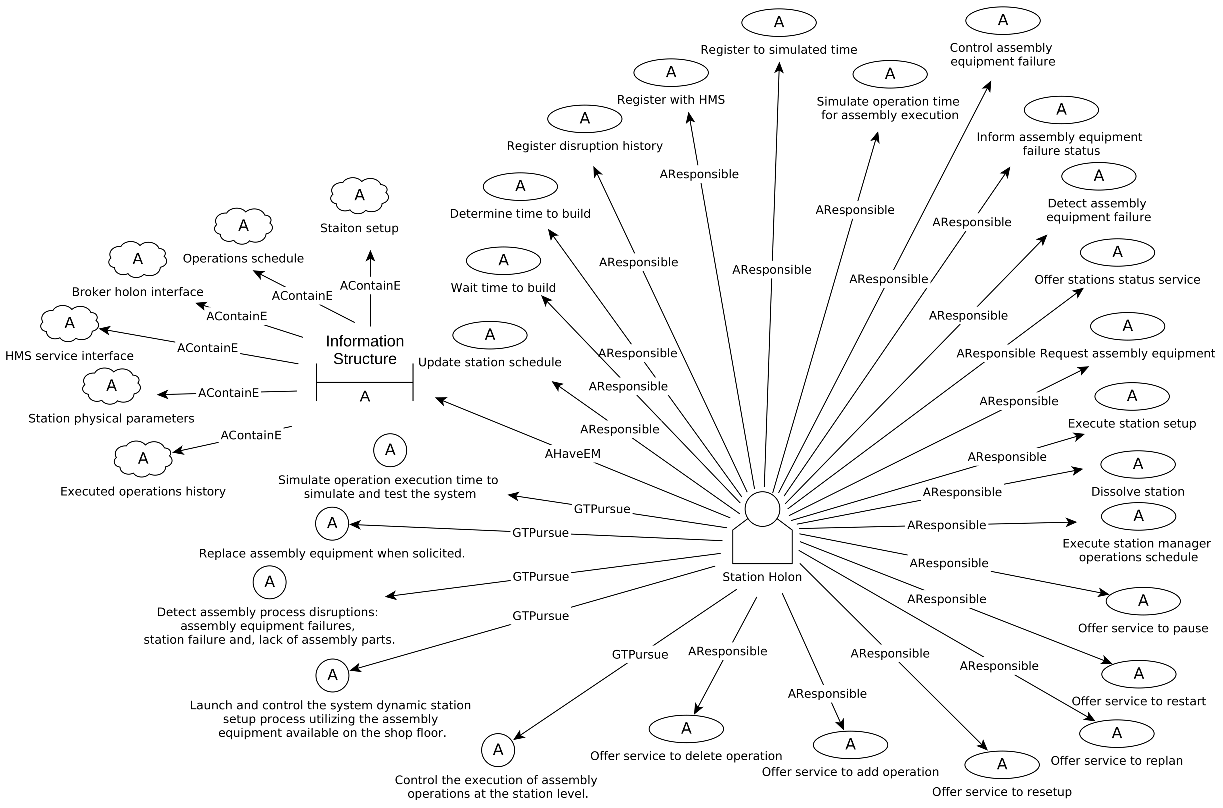

A
Register to simulated time
Control assembly equipment failure
Register with HMS
Simulate operation time for assembly execution
Register disruption history
Inform assembly equipment failure status
Determine time to build
Detect assembly equipment failure
Staiton setup
Operations schedule
Offer stations status service
Broker holon interface
Wait time to build
HMS service interface
Information Structure
Update station schedule
Request assembly equipment
Station physical parameters
Execute station setup
Executed operations history
Simulate operation execution time to simulate and test the system
Dissolve station
Replace assembly equipment when solicited.
Execute station manager operations schedule
Station Holon
Detect assembly process disruptions: assembly equipment failures, station failure and, lack of assembly parts.
Offer service to pause
Launch and control the system dynamic station setup process utilizing the assembly equipment available on the shop floor.
Offer service to restart
Offer service to replan
Offer service to delete operation
Offer service to add operation
Control the execution of assembly operations at the station level.
Offer service to resetup
AResponsible
AContainE
AHaveEM
GTPursue

Figure 32 – Station holon agent diagram.

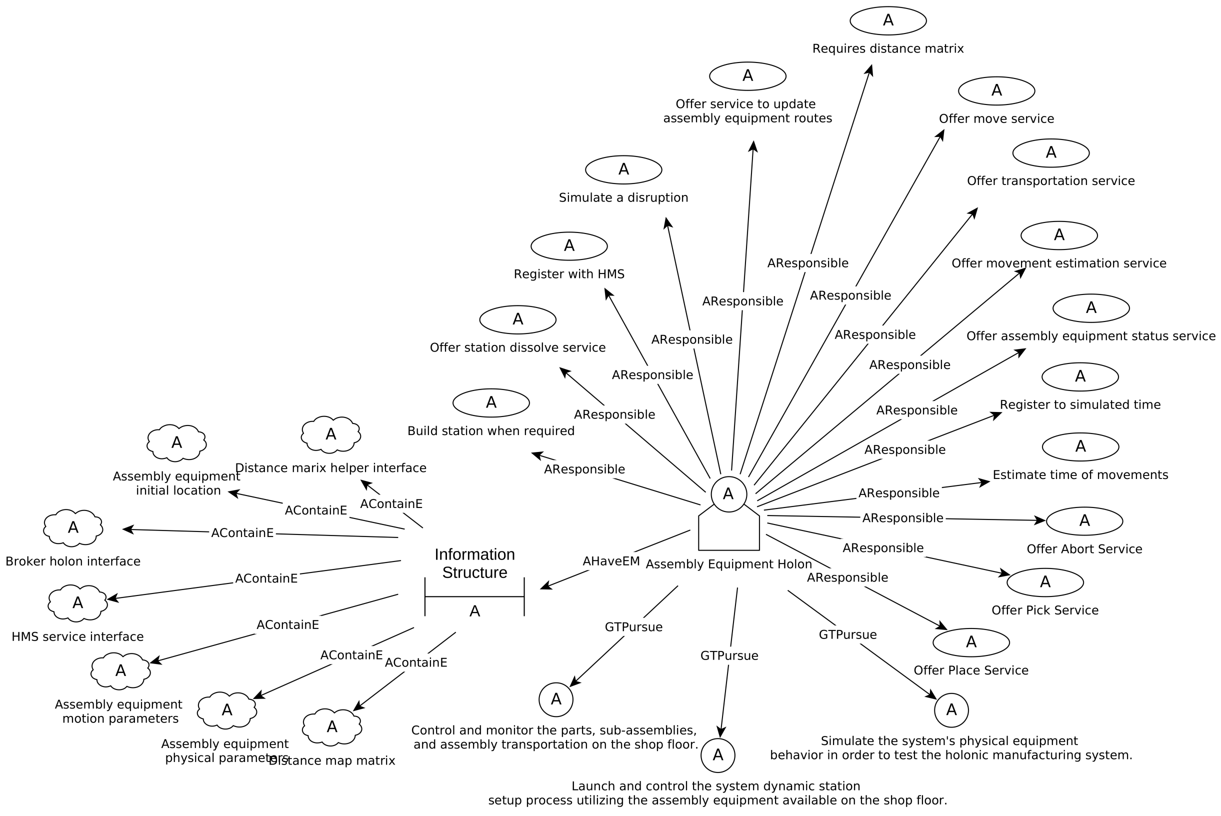


Figure 33 – Assembly equipment holon agent diagram.

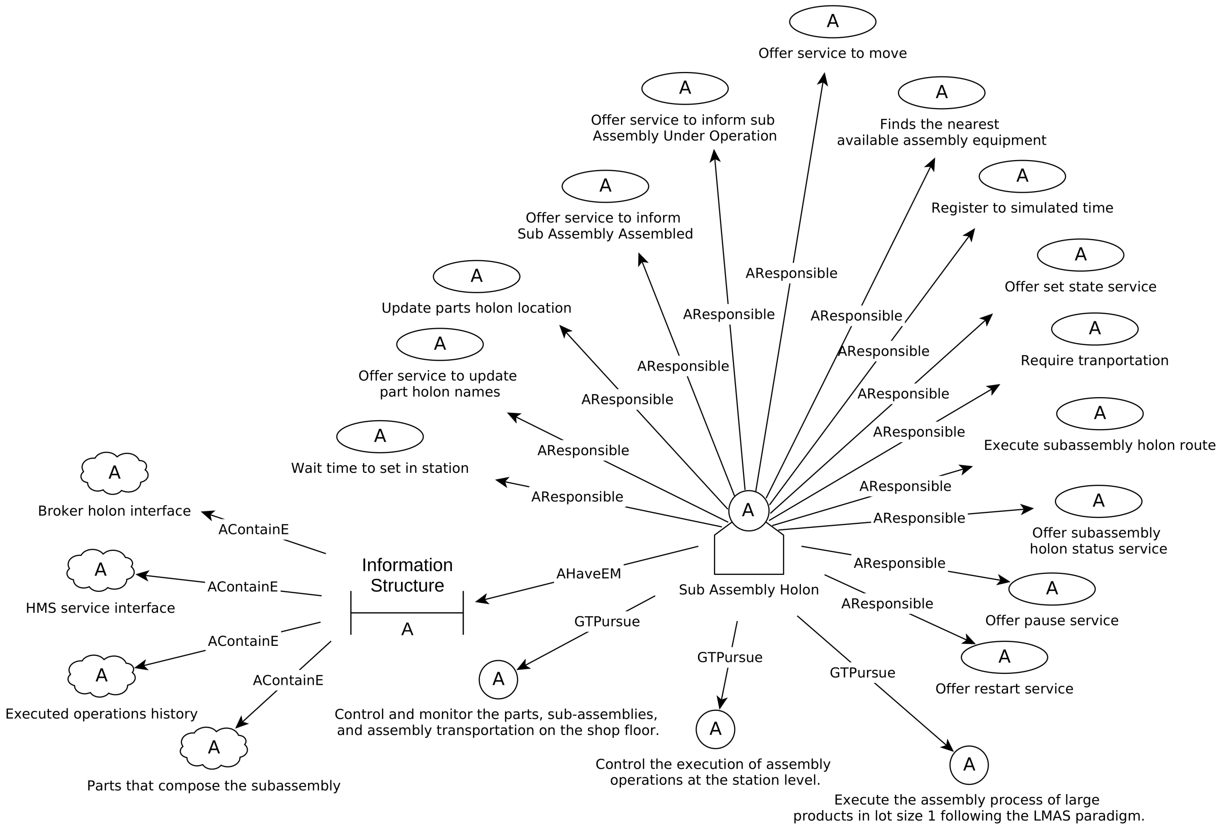


Figure 34 – Sub-assembly holon agent diagram.

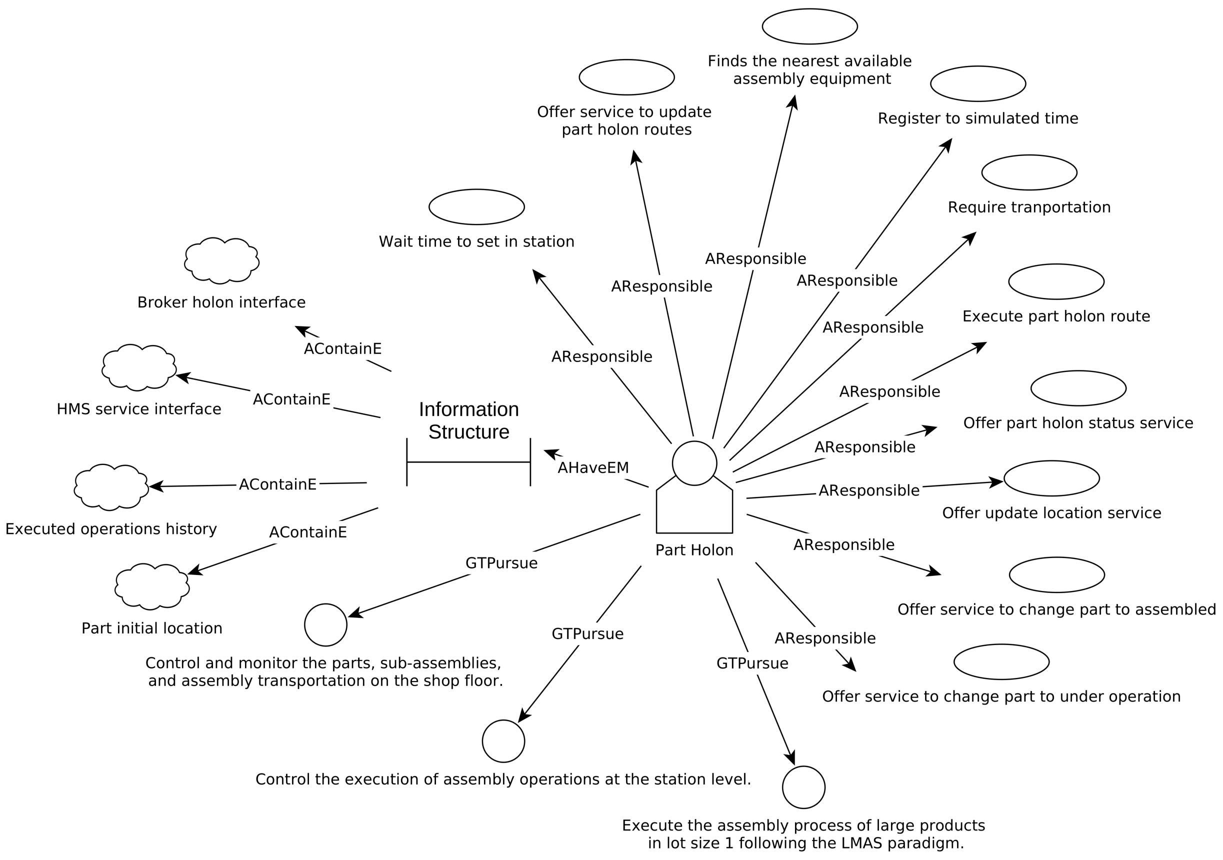

Finds the nearest available assembly equipment
Offer service to update part holon routes
Register to simulated time
Require tranportation
Wait time to set in station
Broker holon interface
Execute part holon route
HMS service interface
Information Structure
Offer part holon status service
Executed operations history
Offer update location service
Part Holon
Part initial location
Offer service to change part to assembled
Control and monitor the parts, sub-assemblies, and assembly transportation on the shop floor.
Offer service to change part to under operation
Control the execution of assembly operations at the station level.
Execute the assembly process of large products in lot size 1 following the LMAS paradigm.
AResponsible
AContainE
AHaveEM
GTPursue

Figure 35 – Part holon agent diagram.

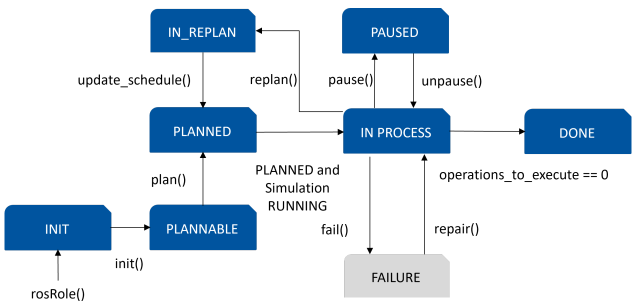
5.2.4 Agent state models

The figure 36 presents a compilation of the operational holons state models. The figure 36a presents the order holon state model. The order holon states are:

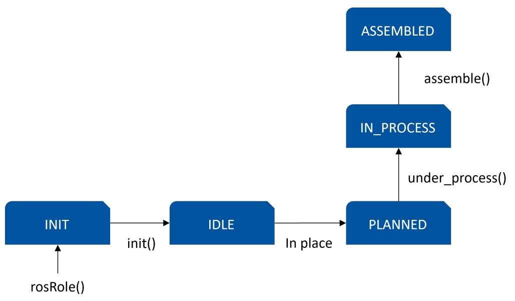
- INIT: entry Point for the order. A logical order object is generated;
- PLANNED: a Plan for the order has been generated;
- PLANNABLE: the order has no schedule;
- IN_PROCESS: order is being executed;
- PAUSED: order is on hold;
- DONE: order is finished;
- FAILURE: state to represent a generic failure;
- IN_REPLAN: order received a replan request. The order waits for an update schedule to set the state to PLANNED.

The order holon state model transition functions are:

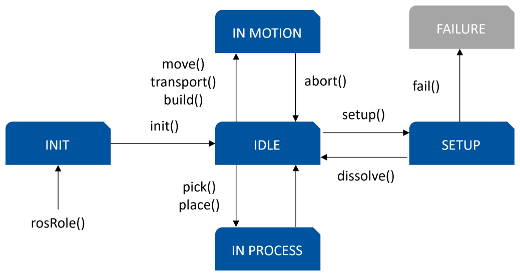
- rosRole(): this is the entryPoint function;



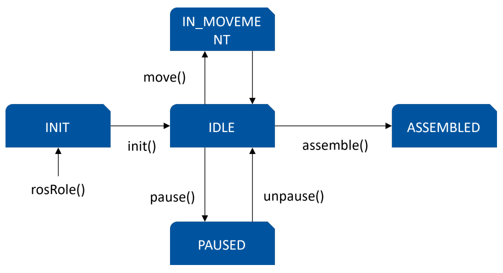
(a) Order holon state model.



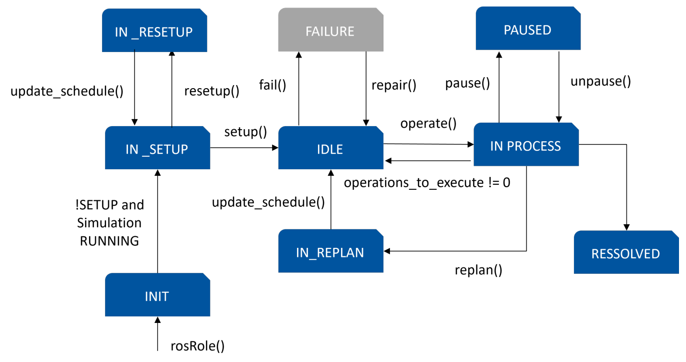
(b) Part holon state model.



(c) Assembly equipment holon state model.



(d) Sub-assembly holon state model.



(e) Station holon state model.

Figure 36 – Operational holons state models.

- init(): internal holon configuration. Set up of provided services and data input parsing. After initializing, the order holon assumes the PLANNABLE state, meaning that a schedule can be sent to the order holon;
- plan(): when receiving a new schedule, the order holon assumes the PLANNED state; PLANNED and simulation RUNNING: the order holon should be in PLANNED and the simulation in RUNNING to start the execution of the operations;
- pause(): the operator sends a command to pause an operation execution;
- unpause(): operator send a command to unpause the execution;
- fail(): a fail detected in one of the stations involved in the execution changes the order holon state to FAILURE state;
- repair(): a repaired confirmation coming from a station brings the order holon back to IN PROCESS state;

- update_schedule(): the order holon received a new schedule from the operator. The order holon now is ready to start again;
- replan(): the order holon received a replan command from the operator. A replan operation is usually sent when disruptions are detected, so the operator can send new schedules aiming to compensate (not part of this dissertation).

The figure 36e presents the station holon state model. The station holon states are:

- INIT: entry Point for the Station. Logical Station object is generated;
- IN_SETUP: station requires resources. They need to be setup;
- IDLE: no operation in station happening, but there is the capacity;
- IN_PROCESS: operation is being executed;
- PAUSED: operation is on hold;
- RESOLVED: station resolves assembly equipment and is no longer operable;
- FAILURE: state to represent a generic failure;
- IN_REPLAN: station received a replan request. In that state the station waits for an update schedule to set the state to IDLE.

The station holon state model transition functions are:

- rosRole(): this is the entryPoint function;
- !SETUP and Simulation RUNNING: If the station is not setup and the simulation is in RUNNING state, then the station holon assumes the IN_SETUP state;
- operate(): once in IDLE, the station holon is able to execute operations;
- setup(): the station holon requires the assembly equipment to come to the station location and execute the station setup;
- pause(): the operator sends a command to pause an operation execution;
- unpause(): operator sends a command to unpause the execution; operations_to_execute == 0: there is 0 operations to be executed; operations_to_execute != 0: there are still operations to be executed;
- fail(): a failure in one of the station's assembly equipment is detected;
- repair(): the station replaced the defective assembly equipment;
- resetup(): the station setup is interrupted by the operator due to a failure in one of the assembly equipment;
- update_schedule(): the station holon received a station containing the operations and assembly equipment involved in the setup process;

- resetup(): a new schedule containing the operations and the assembly equipment for setup is sent to the station holon.

The figure 36c presents the assembly equipment holon state model. The assembly equipment states are:

- INIT: entry Point for the resource;
- IN_MOTION: resource is in motion;
- IDLE: resource is ready to execute operation or move;
- SETUP: resource is successfully allocated in a station configuration;
- IN_PROCESS: operation is being executed using the resource;
- FAILURE: state to represent a generic failure.

The assembly equipment holon state model transition functions are:

- rosRole(): this is the entryPoint function;
- init(): internal holon configuration. Setup of provided services and data input parsing;
- move(): the assembly equipment holon received a move request;
- transport(): the assembly equipment holon received a transport request;
- build(): the assembly equipment received a build station request;
- setup(): the assembly equipment is setup in the station location;
- dissolve(): the assembly equipment holon received a dissolve request from the station holon;
- abort(): the assembly equipment received an abort request from the operator.

The figure 36d presents the sub-assembly holon state model. The sub-assembly holon states are:

- INIT: entry Point for the part;
- IDLE: sub assembly is waiting for simulation to start;
- IN_MOVEMENT: sub assembly is moving;
- PAUSED: sub assembly is pause;
- ASSEMBLED: the assembly of the sub assembly is finished.

The sub-assembly holon state model transition functions are:

- rosRole(): this is the entryPoint function;
- init(): internal holon configuration. Setup of provided services and data input parsing;

- pause(): the operator sends a command to pause the sub-assembly holon;
- unpause(): operator send a command to unpause the sub-assembly holon;
- move(): the sub-assembly holon received a move request;
- assemble(): the sub-assembly is informed that the assembly operation is finished.

The figure 36b presents the part holon state model. The part holon states are:

- INIT: entry Point for the part;
- IDLE: part is waiting for simulation to start;
- PLANNED: part is in place waiting for an operation;
- IN_PROCESS: part is under process;
- ASSEMBLED: the assembly of the part is finished.

The part holon state model transition functions are:

- rosRole(): this is the entryPoint function;
- init(): internal holon configuration. Setup of provided services and data input parsing;
- In place: the part holon is considered in place (in the station) since the beginning of the simulation. the part holon is already considered in place;
- under_process(): the part holon is informed that operation is being executed using the part;
- assemble(): the part holon is informed that the part is assembled to a sub-assembly holon.

5.2.5 Interaction model

In the next subsections, the main interaction models are presented. Among them are included: the send customer order, request optimization, configure holons, and order execution relating to the normal execution (UC1) (c.f. figure 12) and the disruption handling related to disturbances situations (UC2) (c.f. figure 13). The diagrams related to UC1 are presented in the correct order, showing the actions that follow a customer order input.

The lower part of the interaction diagrams always show the sequence of events and highlights the precedence among the interactions, while the upper represents the holons involved, interaction units, and main functions involved in the communication.

5.2.5.1 Send customer order

The figure 37 presents the send customer order interaction diagram. It shows the communication behavior when an operator sends a customer order input to the system specifying the assembly type quantities. The customer order input format is described in the implementation section.

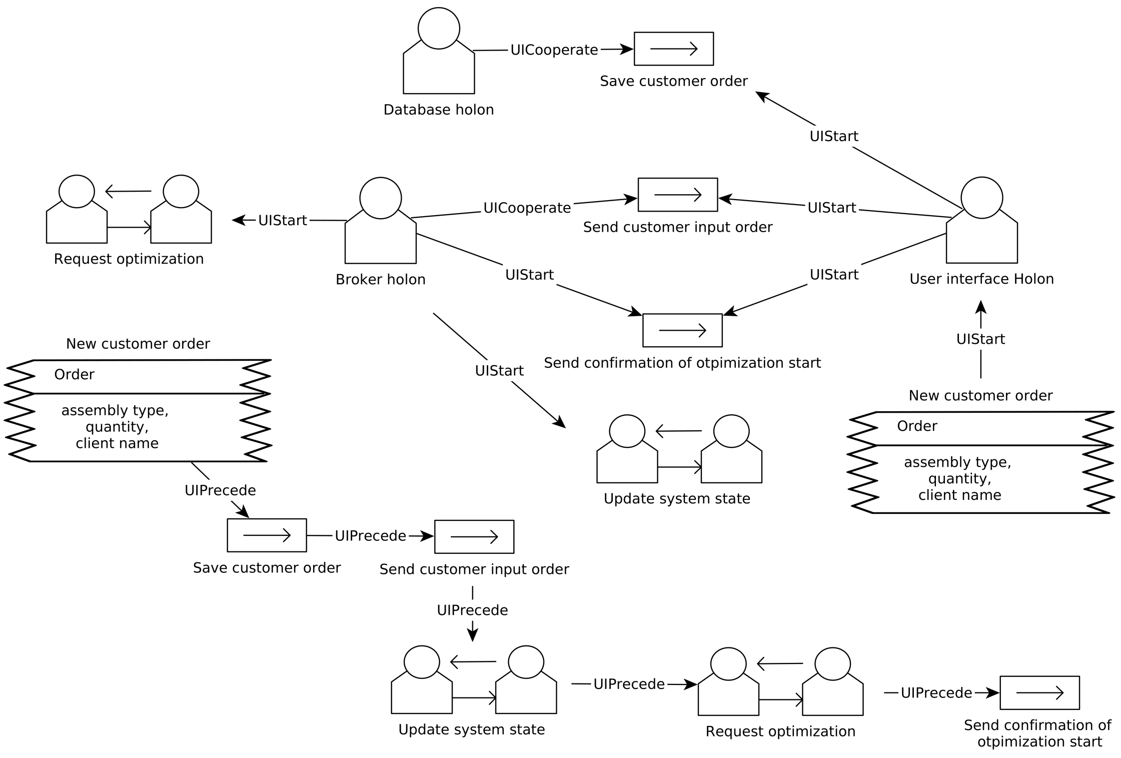


Figure 37 – Send customer order interaction diagram.

5.2.5.2 Request optimization

In figure 38 presents the request optimization interaction diagram. It exhibits the interaction executed by the system in order to enable the broker holon to request a system state snapshot, parse the input data, and request an optimal schedule to the scheduler holon.

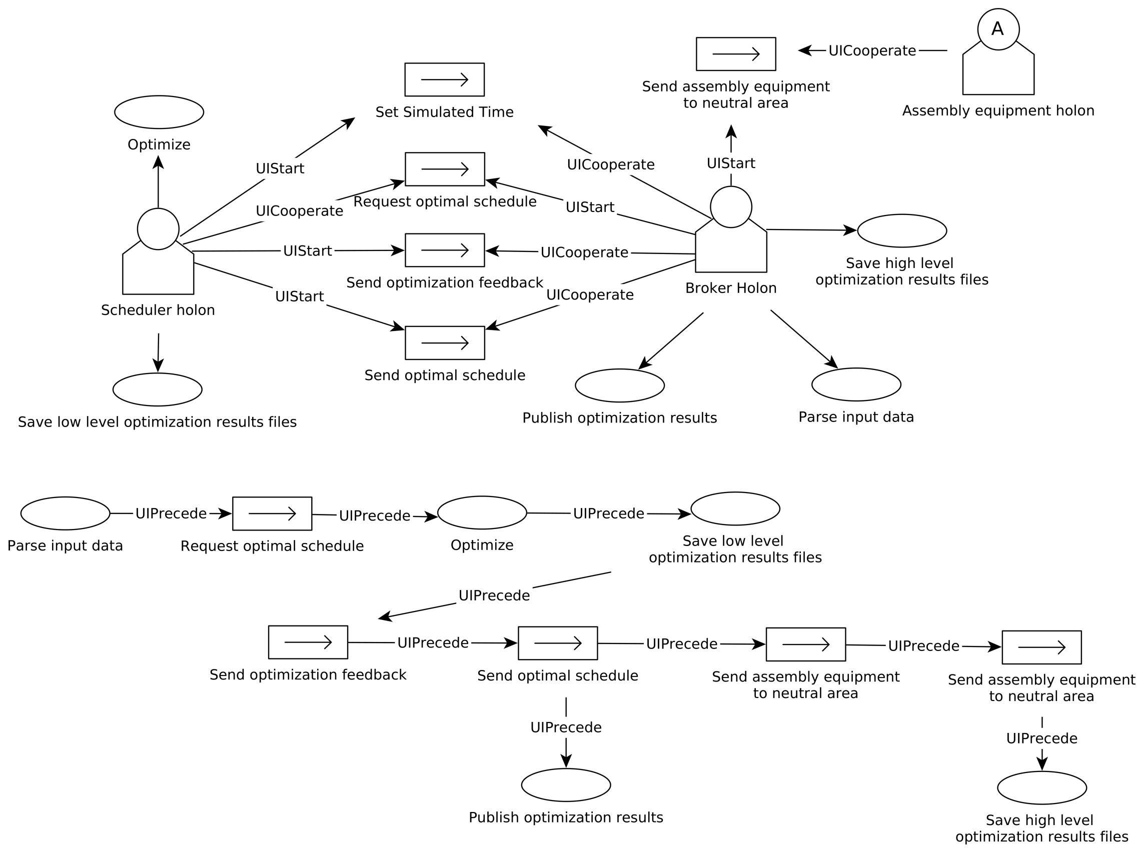

A
Assembly equipment holon
UICooperate
Send assembly equipment to neutral area
UIStart
Set Simulated Time
UICooperate
Optimize
UIStart
Request optimal schedule
UICooperate
UIStart
Send optimization feedback
UIStart
UICooperate
UIStart
UICooperate
Send optimal schedule
Scheduler holon
Broker Holon
Save high level optimization results files
Save low level optimization results files
Publish optimization results
Parse input data
Parse input data
UIPrecede
Request optimal schedule
UIPrecede
Optimize
UIPrecede
Save low level optimization results files
UIPrecede
Send optimization feedback
UIPrecede
Send optimal schedule
UIPrecede
Send assembly equipment to neutral area
UIPrecede
Send assembly equipment to neutral area
UIPrecede
Publish optimization results
UIPrecede
Save high level optimization results files

Figure 38 – Request optimization interaction diagram.

5.2.5.3 Configure holons

in the figure 39 the configure holons interaction diagram is presented. It shows the procedures executed by the broker holon in order to distribute the optimal schedule for the five operational holons: assembly, assembly equipment, order, station and part holons.

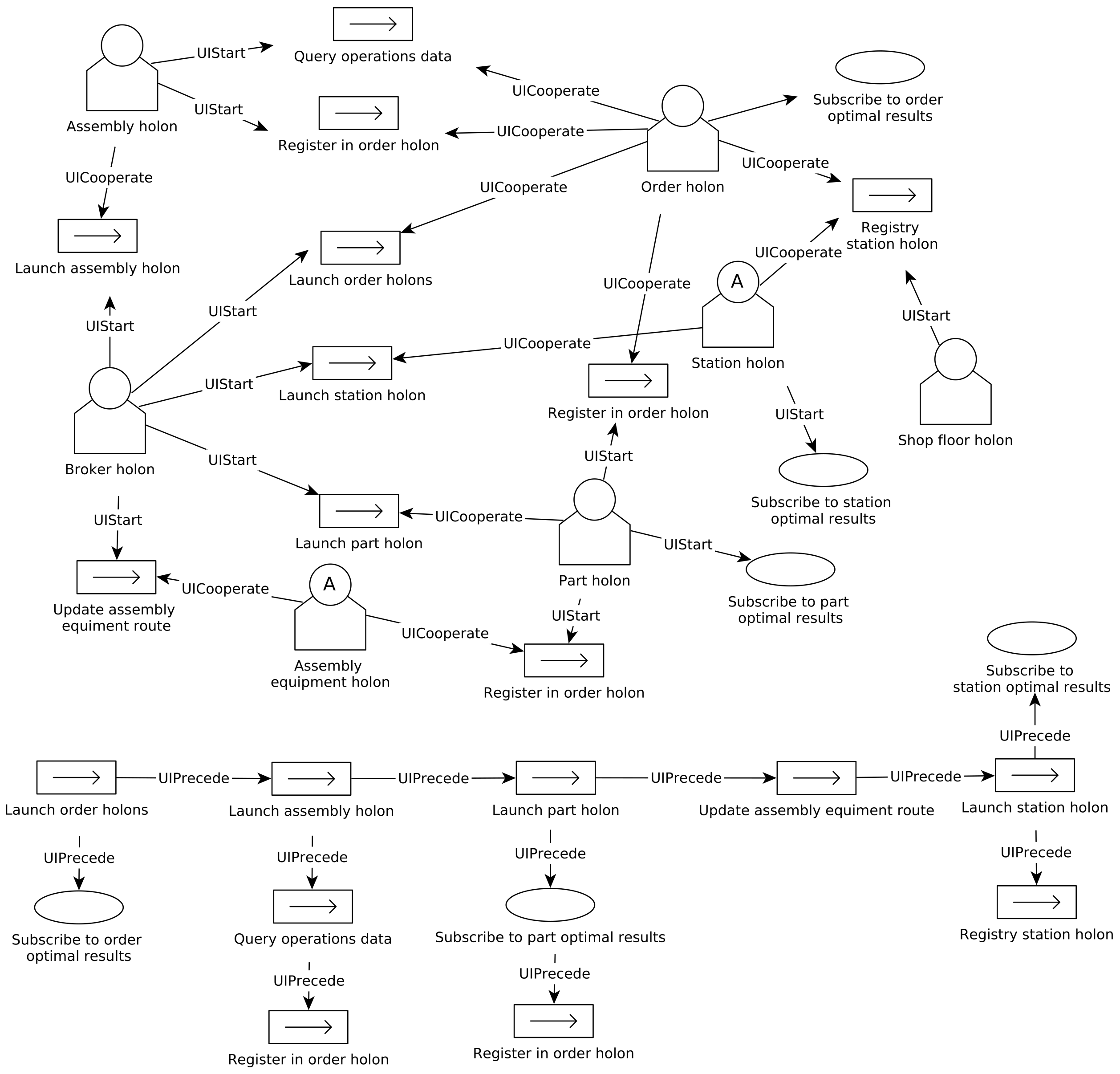


Figure 39 – Configure holons interaction diagram.

5.2.5.4 Order execution

In figure 40 the order execution interaction diagram is presented. It shows an intricate interaction behavior among the order, station, sub-assembly and part holons. This interaction represents the operation execution process until the order holon executes all the pending operations provided in the optimal plan. The basic idea is that the order holon waits for the station, sub-assembly, and part holons to be in place and ready for execution. After these three conditions are fulfilled, the order holon sends an execute operation command to the station. If the sub-assembly holon is not in place, the order holon requests the sub-assembly to move to a specific location while the part holons are considered already in place (see simplifications and assumptions, table 6).

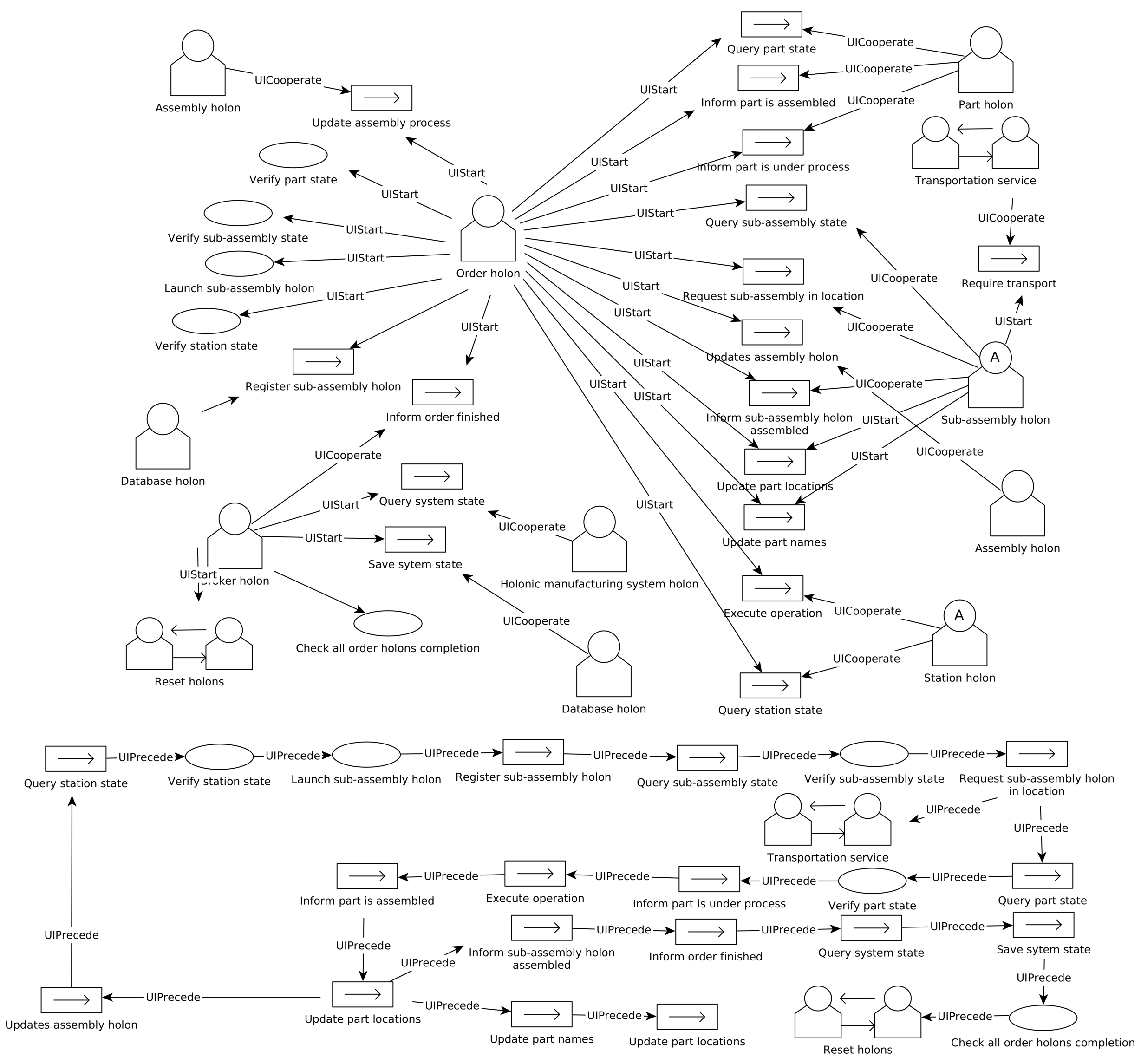


Figure 40 – Order execution interaction diagram.

5.2.5.5 Disruption handling

Figure 41 represents the disruption handling interaction diagram. It shows the interactions executed by the switch mechanism, order and assembly equipment holons in order to provide a corrective behavior to solve assembly equipment disruptions. The only disruption covered is the assembly equipment failure, while this assembly equipment is SETUP state in the station (see simplifications and assumptions table, 6). Two solutions are provided, so two different linear lines are observed in the lower part this diagram. The local fix solution involves replacing defective assembly equipment for another of the same assembly equipment type. In that case, the station holon is responsible for identifying the assembly equipment among the available equipment. The complete reconfiguration involves the previous identification of executed operations and the later complete reconfiguration of the system, meaning that the whole system undergoes a reset and new order holons are triggered considering the remaining operations to be executed.

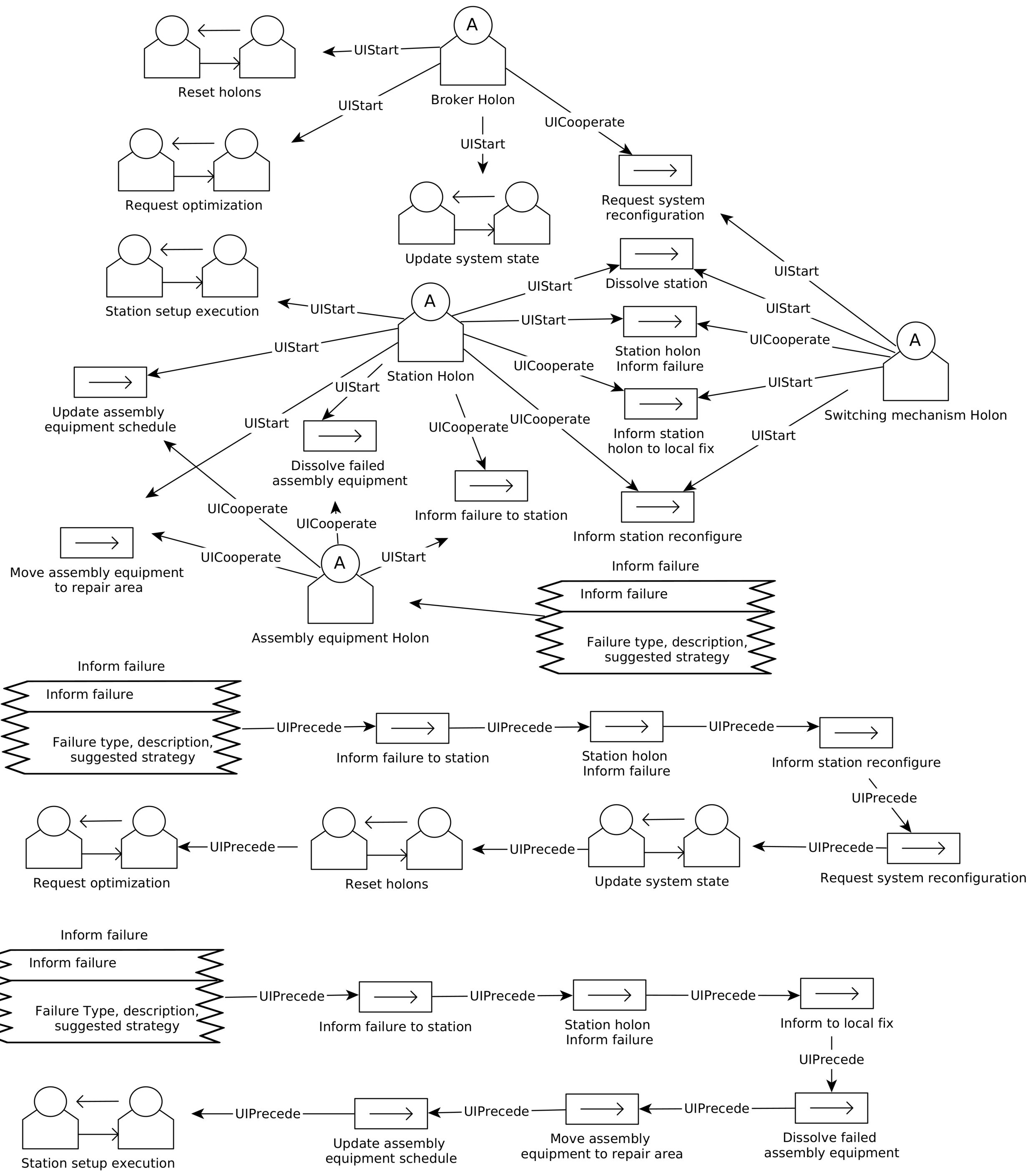

Reset holons
Broker Holon
UIStart
UIStart
UIStart
UICooperate
Request optimization
Update system state
Request system reconfiguration
Station setup execution
Dissolve station
Station Holon
Station holon Inform failure
Update assembly equipment schedule
Dissolve failed assembly equipment
Inform station holon to local fix
Switching mechanism Holon
Inform failure to station
Inform station reconfigure
UICooperate
UIStart
Move assembly equipment to repair area
Assembly equipment Holon
Inform failure
Failure type, description, suggested strategy
UIPrecede
Inform to local fix
Failure Type, description, suggested strategy

Figure 41 – Disruption handling interaction diagram.

5.2.6 Architecture models considerations

The compound of the presented models forms the models for the HCA aimed for the LMAS. Its evaluation is executed in qualitative and quantitative formats in chapter 7. In the following section some considerations for its design process are given in the format of question answer:

1. Why is it "good" or "bad" architecture for LMAS?

The presented architecture models are suitable for LMAS because: at a low-level perspective, they address the individual order routes (figure 40), the required dynamic station setups, and disturbance handling (figure 41); at a high-level perspective, the models are based on the HMS concept; thus, they represented an HCA implementation that addresses the hierarchical and heterarchical behavior ending up into a holarchy of holons. The LMAS, for stability reasons, requires a hierarchical approach and can't rely on a fully automated and heterarchical approach that might end up in a chaotic behavior.

However, the proposed architecture models don't present a intricate mechanism to deal with disruptions and decision-making. Instead, the models offer a ground basis for an LMAS implementation and can be seen as a template to which various disruption handling strategies could be attached.

2. How does it differ from other implementations?

The architecture presented in figure 29 relates to reviewed literature (see more in chapter 4) in the following points:

- it presents a clear distinction between the roles of holons into three layers. Related work usually represents the roles within the definition of the holon itself and does not explicitly specify a common line of action for holons, at least in terms of goals;
- the holon naming and roles are based on those high-level holonic classes defined in the PROSA architecture (Product, Resource, Order and Staff);
- the interaction and agent models consider the presence of a switch mechanism holon responsible for centralizing the decision making when the system is handling a disruption. The idea of the switch mechanism holon is presented in the literature, and it is useful for system modularization and to concentrate the strategies for disruption handling in one entity. The switch mechanism holon implementation is rooted in the specialization allowed by the Staff holon proposed by the PROSA architecture;
- it is an ontology-driven HCA (Holontology at the right part of the figure 29). In other words, the concepts, interactions, and predicates utilized in the agents and their interactions were based on an ontology, namely Holontology. Holontology

is also part of this work, and its process of development is described later in chapter 6. The idea of a holonic ontology embracing HMS concepts is presented in the literature, but no HCA was explicitly derived from one;

- on its interaction model, the architecture explicitly presents the interactions necessary to form stations in the factory shop floor dynamically. In the literature, the holonic aggregation is shown but not oriented to address individual resources and form stations regardless of time and location. In other words, the working stations can be formed at any time and place, using the required assembly equipment to setup.

3. Why is it flexible?

The presented models are flexible mainly because of the disruption handling presented in figure 41. According to the behavior described in this diagram, the station and the switch mechanism holons can handle a system disruption in two different manners: LOCAL_FIX and REORGANIZATION. This model adds up to the systems, once hierarchical approach, a heterarchical characteristic, bending the models towards the flexible approach. Both hierarchical and hierarchical methods, when merged, give birth to the holarchy structure, giving the system the ability to react to disruptions organizing the system in a hierarchical or heterarchical manner.

6 System architecture implementation

6.1 System data model

During the design of the ANEMONA deployment models, the guidelines suggest the use of ontologies as the main structure for the messages exchanged by the holons. Following this approach, an ontology study was carried out to understand whether a new ontology had to be derived or the re-use of ontology models could be executed. Therefore this chapter covers the implementation aspects of the Holontology (an ontology used as a basis for the interactions and holons) and the data model that is derived from it to support the HCA implementation; the factors considered for the decision to use ROS as the framework to implement the HCA prototype; The HCA ROS Templates, which characterize the main holons' functions, data and behavior; and the packages implemented using the ROS framework.

6.1.1 Ontology

To determine whether it is necessary to develop a whole ontology for the holonic architecture or even to start from an ontology preconception, typical manufacturing ontologies were studied MASON (LEMAIGNAN *et al.*, 2006), MSDL (AMERI; DUTTA, 2006), MaRCO (JÄRVENPÄÄ *et al.*, 2019) and BaSys (WESER; BOCK, 2020). The software Protègè was used to query the ontologies and verify whether the principles of LMAS can be fulfilled, the ontologies used in that study were modeled with OWL descriptive language, which is interpreted by the Protègè software. The queries work as questions that expect "yes" or "no" as answers. In the methodology applied in this work, in the case of a "yes" answer, the answer should be accompanied by an example. The questions used to evaluate the presented ontologies were inspired in the ontology requirements presented in table 4:

(a) Is there the capacity to execute a transport? (Transport)

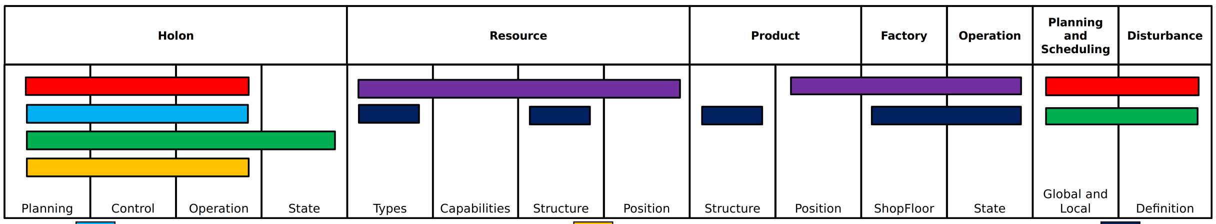
(b) Is there the capacity to move assembly equipment/resources to a specific place? (Mobile resources)

(c) Is it possible to group a set of entities capable of executing assembly tasks? With a temporal character? (Temporary stations)

(d) Is there the possibility to assign schedules to resources? (Individual order routes)

(e) Is it possible to represent physical places of the shop floor? (Organization levels)

The questions presented are human-readable, though it was necessary to adapt them to each ontology terminology, which required a comprehensive study about the



🟦 PROSA(BRUSSEL *et al.*, 1998) 🟨 Ling Gou et al.(GOU *et al.*, 1998) ⬛ MASON(LEMAIGNAN *et al.*, 2006) 🟥 ADACOR(2007)(BORGO; LEITÃO, 2004) 🟩 Marmolejo at al.(SIMóN-MARMOLEJO *et al.*, 2018) 🟪 MaRCO(JÄRVENPÄÄ *et al.*, 2019)

Figure 42 – LMAS Data Model Merging Process

ontologies semantic, to generate adapted inputs to the Protègè query mechanism. Table 8 shows the evaluation results. The results indicate that none of the analyzed ontologies fully implemented the necessary LMAS principles; thus the definition of a new ontology is executed.

Table 8 – Ontology evaluation results.

Query	MASON	MSDL	BaSyS	MaRCO
(a)	Handling	Stock Material	C2Transport	Transporting TransportingCollaborative
(b)	-	-	C2 MoveToPos	Moving MovingCollaborative
(c)	-	-	-	DeviceCombination RealDeviceCombination Station
(d)	Human Operation	-	-	-
(e)	Shop Floor	-	-	Area/Site/Cell

The new ontology, called Holontology, is proposed to draw physical and cyber concepts of the manufacturing domain. The separation between physical and cyber parts of this ontological approach is motivated by the definition of the basic unit, the holon. Following the methodology from (LEGAT *et al.*, 2014), the first three steps of the methodology consist of selecting informational aspects (first row of figure 42), identifying the informational aspects' requirements (last row of figure 42), and selecting ontology modules (legend of figure 42). The remaining phases concern ontological alignment and ontology development, the alignment and the merging process is represented in the diagram in figure 42.

The alignment process aims to identify the similarities among the concepts described in the selected ontologies to merge them. The ontology development consists of selecting the concepts necessary to fulfill the LMAS requirements and implementing the FLExible Resource Manufacturing Ontology (FLERMO), and the HOLonic assembLy manufacturing ONTOlogy (HOLLONTO). The FLERMO consists of concepts related to the physical world, and is inspired on MASON (LEMAIGNAN *et al.*, 2006) and MaRCO (JÄRVENPÄÄ *et al.*, 2019) while the HOLLONTO comprises the holonic cyber concepts (GOU *et al.*, 1998; BRUSSEL *et al.*, 1998; LEITÃO; RESTIVO, 2006; SIMóN-MARMOLEJO *et al.*, 2018). The figure 43 summarizes the procedure applied for the LMAS data model implementation.

6.1.2 Data model

The Holontology is created through the merge and fine selection of concepts defined in HOLLONTO and FLERMO. From the Holontology essential concepts and data properties are extracted to construct the data model used in the LMAS HCA architecture described in the chapter 5. A high-level comprehension of the data model is shown in figure 44.

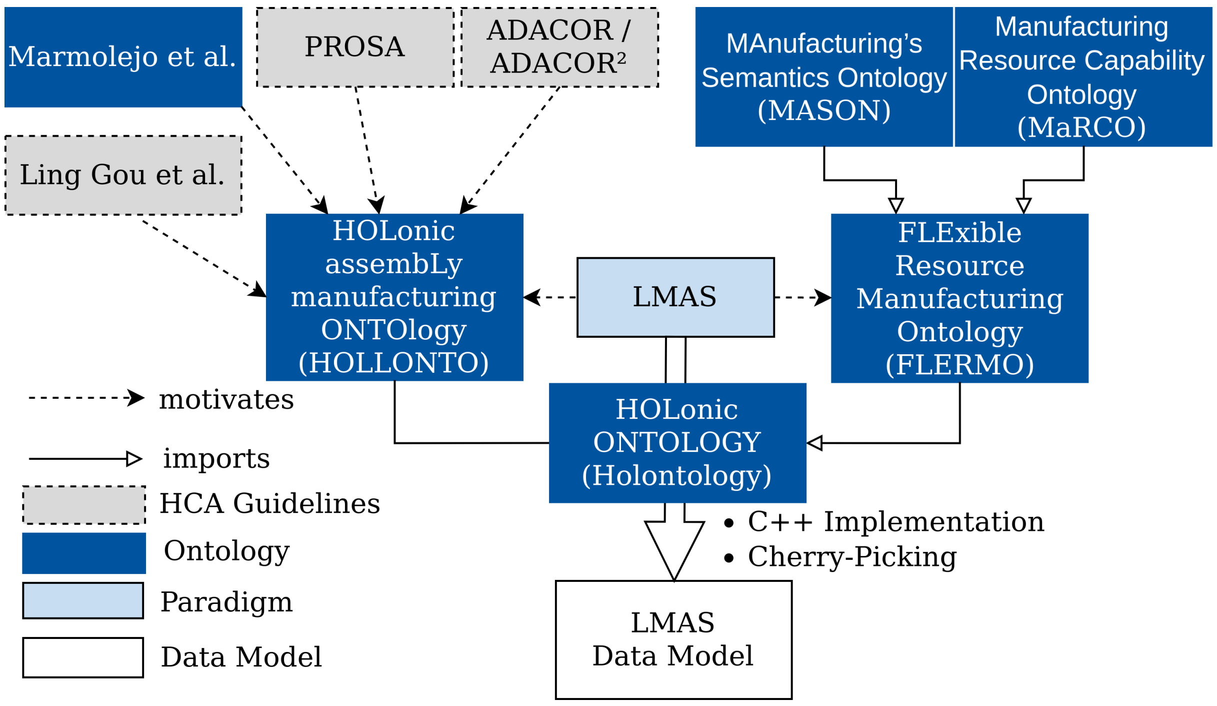

Marmolejo et al.
PROSA
ADACOR / ADACOR²
MAnufacturing's Semantics Ontology (MASON)
Manufacturing Resource Capability Ontology (MaRCO)
Ling Gou et al.
HOLonic assembLy manufacturing ONTOlogy (HOLLONTO)
LMAS
FLExible Resource Manufacturing Ontology (FLERMO)
HOLonic ONTOLOGY (Holontology)
• C++ Implementation
• Cherry-Picking
LMAS Data Model
motivates
imports
HCA Guidelines
Ontology
Paradigm
Data Model

Figure 43 – Holontology: Ontologies Reuse Method

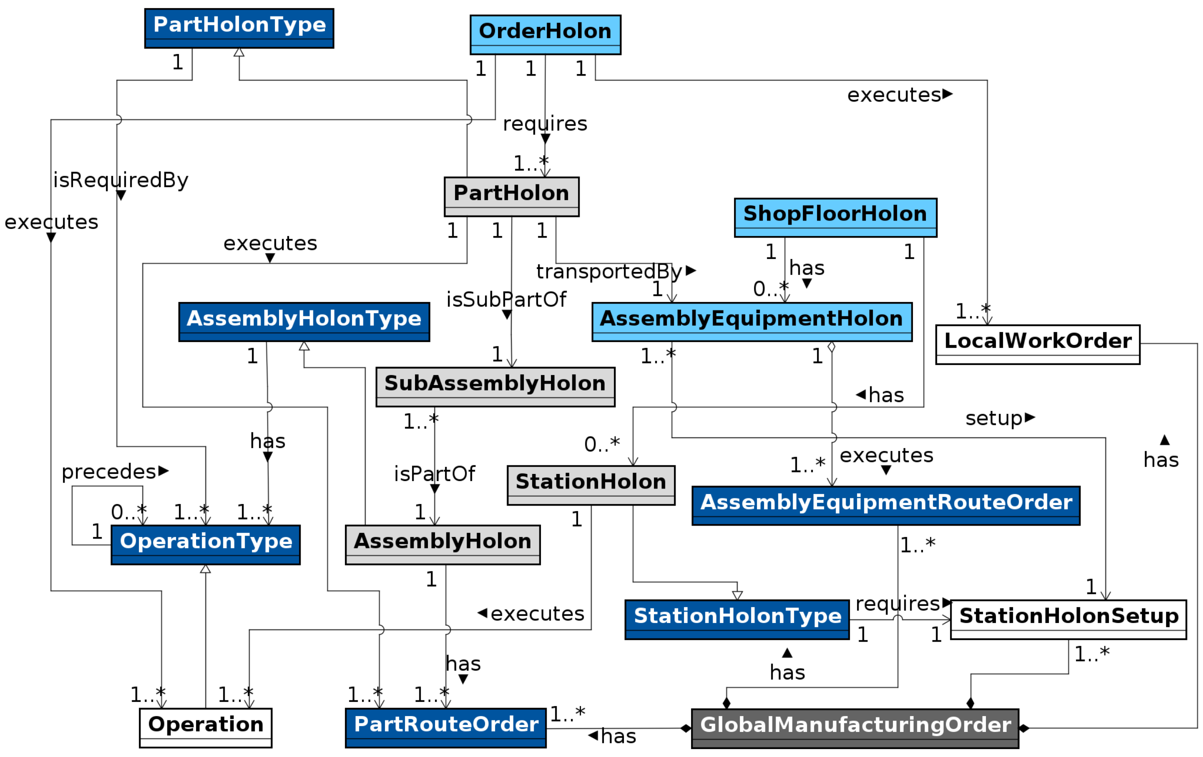


Figure 44 – LMAS HCA Data Model (Colors refer to Conceptual Similarities) 🟦 PROSA(BRUSSEL *et al.*, 1998) ◻ Ling Gou et al.(GOU *et al.*, 1998) □ ADACOR(2007)(BORGO; LEITÃO, 2004) ■ Marmolejo at al.(SIMóN-MARMOLEJO *et al.*, 2018) 🟦 Authors

6.2 Robotic programming framework

Table 9 – Framework suitability analysis for Line-Less Mobile Assembly Systems. (++ fully applied; + partially applied; 0 not applied)

Framework	(a)	(b)	(c)	(d)	(e)	(f)	(g)
JADE	+	0	+	++	++	++	++
ROS	++	++	++	+	0	++	++
SPADE	++	0	+	++	++	+	0
BaSys 4.0	+	0	+	0	0	0	0
JaCaMo	0	0	+	++	0	++	0

A programming framework shall be selected for the HCA implementation. Generally, multiple frameworks are available (IÑIGO-BLASCO *et al.*, 2012; PAL *et al.*, 2020) offering the necessary capabilities compliant with the LMAS requirements. A collection of frameworks was elected for further analysis. For the analysis it was considered the following aspects:

(a) robotics algorithms availability: the LMAS lower-level layer should be prepared to deal with robotic systems. In that case, the availability of robotic software modules and their reusability is a plus on the analysis;

(b) robotic simulation tools: it is mandatory to simulate the environment to check the assertiveness of the holonic control architecture and the LMAS optimization module. It is a plus if the framework presents a long-term compatibility/integration with a simulation tool.

(c) programming language suitability: most robotics applications use, as a programming language, C++ due to performance requirements. (IÑIGO-BLASCO *et al.*, 2012). If the framework includes C++ language support, it is a plus;

(d) mechanism to act with social abilities/agent behavior: as stated by (IÑIGO-BLASCO *et al.*, 2012; GIRET *et al.*, 2017) an agent framework exhibiting agent social abilities is one of the technology enablers to develop MAS/HMS systems. If the framework provides a protocol based on messages, topics, services, or even a higher level of abstraction for speech acts, it is a plus;

(e) FIPA compliant: the Foundation for Intelligent Physical Agents (FIPA) offers a set of standards envisioning the improvement of interoperability between agent-based solutions. If the protocol is structured following the FIPA message structure and message exchange protocol, it is a plus;

(f) research community usage: this aspect represents a qualitative author's evaluation of community framework usage. The community usage and activity are mandatory for technical support and discussions;

(g) industry usage/acceptance: this aspect comprises the industry usage. It is a plus if the framework has an acceptance and organized groups that study and encourage using it for real industry applications. The industrial acceptance analysis for ROS, JADE and JaCaMo are based on (ROS-INDUSTRIAL, 2020), (BELLIFEMINE *et al.*, 2008) and (BOISSIER *et al.*, 2020) respectively;

As depicted in table 9 the ROS framework presents suitability according to the selected criteria (a-g), and it will be used as a framework to implement the holonic architecture along with the data model and other LMAS modules. Important to mention is that the BaSys framework (BASYS, 2021) presents an exciting alternative for Industry 4.0 interoperability.

6.3 System platforms and ROS templates

On the first step of the ANEMONA "build architecture" phase, the architect must specify the number of agent platforms to distribute the application. The distribution criteria are based on the following guidelines:

- different departments/branches may require a unique platform for each, resulting in a multi-platform system;
- if the number of abstract agents is not excessively high and the holons interactions do not exceed a specific cooperation context, the system may be implemented on a single platform;
- security, data-encapsulation, and functionality requirements may be reasons for choosing a distributed system instead of a single platform, especially when the number of agents is high.

The table 10 shows the resulting distribution. The database holon has its platform because of data security and modularity reasons. The scheduler holon has its platform due to functionality and performance requirements. If the optimization algorithm is running on the same platform, the performance of the overall HCA has been heavily affected accordingly to practical experiences. The effects are the result of the significant computational power required to execute the optimization. It impacts ROS communication performance and potentially leads to unexpected delays and loss of communication (timeout).

The control execution platform contains the station, switch mechanism, subassembly, and order holons. These holons are digital-only holons, which act as conductors of the assembly execution on the shop floor. They monitor and control the operational holons and do not perform heavy tasks (e.g., image processing; optimization algorithms).

The assembly equipment holons are designed to have an individual platform. Individualization is inspired and characterized by the fundamental nature of resources utilized in manufacturing. Each holon can execute a specific task (e.g., pick, place, navigation, image processing, drilling), which requires different mechanical assets and software.

The system knowledge platform contains the holons responsible for tracking the operations and holding the assembly and part physical and dynamic knowledge.

The rest of the holons are grouped on a platform named shop floor due to its low computational requirements. This division groups self-similar or similar-context holons due to the fact of low computational power consumption and their goals.

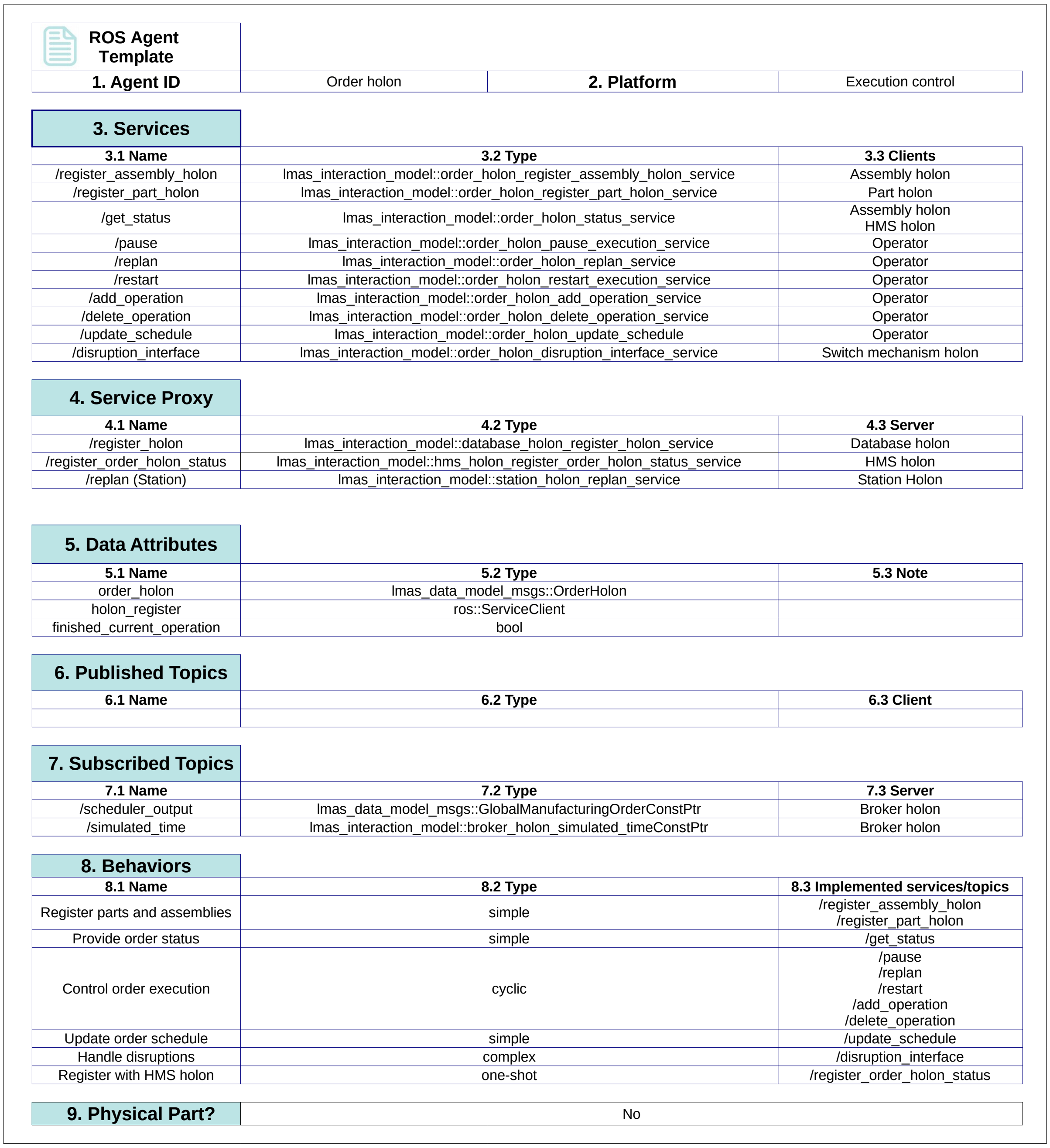
The second step of the ANEMONA "build architecture" phase consists of deriving the JADE (Java Agent DEvelopment Framework) templates for each holon declared in the design phase (see chapter 5). The original JADE template contains: agent identifies, behavior, service, communication, and ontology specification. This template serves as an agent recipe that a software engineer should follow to program the system. As declared in section 6.2 ROS is used instead of JADE. However, the same characteristics of the JADE template were considered when creating the ROS template. An example is depicted in figure 45. It shows the characteristics of the order holon. The ROS template declares:

Table 10 – HCA platforms.

Platform name	Holons
Scheduler	Scheduler holon
Database	Database holon
Shop floor	Broker, Station type, part type, station type, capability, assembly equipment type, shop floor, user interface and holonic manufacturing system holons
Execution control	Station, switch mechanism holon, sub assembly, and order holons
Assembly equipment i	Assembly equipment holon
System knowledge	Assembly and part holons

1. **Agent ID**: contains a unique holon name;
2. **Platform**: contains the platform definition for the described holon;
3. **Services**: contains the services offered by the order holon to the system, their types and their potential clients;
4. **Service Proxy**: this item contains the services used by the holon, their types, and the holons that serve them;
5. **Data attributes**: this item contains data attributes that make part of the system class definition, including the data type derived from the Holontology (see section 6.1);
6. **Published topics**: this section contains the published data topics;
7. **Subscribed topics**: this section contains the subscribed data topics;
8. **Behaviors**: this section has a direct connection to the former JADE template. The designer should specify the holon behaviors by showing the services used to implement them;
9. **Physical part**: this section contains a boolean value declaring whether or not the holon has a physical asset.

The ANEMONA's interaction, agent, and environment models were analyzed to fill the ROS templates for various holons. During the implementation stages, a precise definition of attributes and services' data types was incremented to the ROS templates following the iterative and cyclic characteristics of ANEMONA methodology.



ROS Agent Template

1. Agent ID	Order holon	2. Platform	Execution control

3. Services

3.1 Name	3.2 Type	3.3 Clients
/register_assembly_holon	lmas_interaction_model::order_holon_register_assembly_holon_service	Assembly holon
/register_part_holon	lmas_interaction_model::order_holon_register_part_holon_service	Part holon
/get_status	lmas_interaction_model::order_holon_status_service	Assembly holon HMS holon
/pause	lmas_interaction_model::order_holon_pause_execution_service	Operator
/replan	lmas_interaction_model::order_holon_replan_service	Operator
/restart	lmas_interaction_model::order_holon_restart_execution_service	Operator
/add_operation	lmas_interaction_model::order_holon_add_operation_service	Operator
/delete_operation	lmas_interaction_model::order_holon_delete_operation_service	Operator
/update_schedule	lmas_interaction_model::order_holon_update_schedule	Operator
/disruption_interface	lmas_interaction_model::order_holon_disruption_interface_service	Switch mechanism holon

4. Service Proxy

4.1 Name	4.2 Type	4.3 Server
/register_holon	lmas_interaction_model::database_holon_register_holon_service	Database holon
/register_order_holon_status	lmas_interaction_model::hms_holon_register_order_holon_status_service	HMS holon
/replan (Station)	lmas_interaction_model::station_holon_replan_service	Station Holon

5. Data Attributes

5.1 Name	5.2 Type	5.3 Note
order_holon	lmas_data_model_msgs::OrderHolon	
holon_register	ros::ServiceClient	
finished_current_operation	bool	

6. Published Topics

6.1 Name	6.2 Type	6.3 Client

7. Subscribed Topics

7.1 Name	7.2 Type	7.3 Server
/scheduler_output	lmas_data_model_msgs::GlobalManufacturingOrderConstPtr	Broker holon
/simulated_time	lmas_interaction_model::broker_holon_simulated_timeConstPtr	Broker holon

8. Behaviors

8.1 Name	8.2 Type	8.3 Implemented services/topics
Register parts and assemblies	simple	/register_assembly_holon /register_part_holon
Provide order status	simple	/get_status
Control order execution	cyclic	/pause /replan /restart /add_operation /delete_operation
Update order schedule	simple	/update_schedule
Handle disruptions	complex	/disruption_interface
Register with HMS holon	one-shot	/register_order_holon_status

9. Physical Part?	No

Figure 45 – Order holon template description.

6.4 ROS packages

The ROS development process was based on packages following a functionality-context separation logic to define where the holon's functions should be allocated (e.g., data attributes and interaction). The implementation of the LMAS HCA prototype holons is separated into three ROS sub-packages: role, interaction, and asset packages. The role package contains the message definitions for each holonic class. The LMAS data model is the basis for each message in role classes.

The interaction package contains the functions and auxiliary message definitions to holons' interactions. The functions are defined according to the Holontology's object property. Every holon has an entry-point function designed as rosRole, acting as the entry-point for the holon initialization.

The asset package contains the ROS launch files that initialize the LMAS HCA holarchy. In figure 46, the holon layers are depicted, highlighting the functionality given by each package in the ROS workspace.

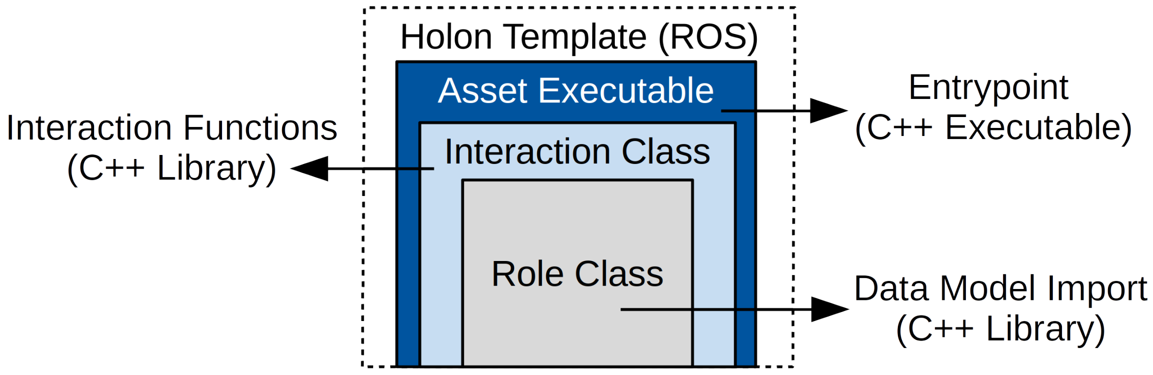

Holon Template (ROS)
Asset Executable
Interaction Class
Role Class
Interaction Functions (C++ Library)
Entrypoint (C++ Executable)
Data Model Import (C++ Library)

Figure 46 – ROS packages for HCA.

6.5 Input files generator

In this section, the structure of the input files is described. The input files represent the necessary information the HCA should hold to interpret the environment, configure the simulation, configure the optimization layer with the correct assembly information and represent the systems limitations and manufacturing capabilities (e.g., processing times).

The structure used on these files follows the JavaScript Object Notation (JSON) notation. Seven files have been proposed. The file names use the terminology described in the data model. The files proposed are:

- part_type.json: the file describes the part type information (e.g., physical and dynamic properties);
- assembly_type.json: file describes the assembly type information. Assembly type information consists of the assembly order, priority graph, the assembly id, the physical and dynamic properties, and the operations;
- assembly_equipment_type.json: the assembly equipment file describes how many different types of resources are available (e.g., AGV_BRAND_1, TOOL_X, FIXTURE_Y);
- station_type.json: the station type file describes the set of possible station configurations the system can manage the assembly equipment to assume. Each station configuration contains the capabilities and the intra-station organization description;

- assembly_equipment.json: the resource files represent the assembly equipment available and the associated resource type. This file can be understood as an instantiation of the assembly equipment types. It represents the number of current resources on the shop floor;
- customer_order.json: the customer order file describes how many different assemblies are requested by a set of customers. The file also informs the earliest start time and latest finish time for each order;
- shop_floor.json: the shop floor file represents the shop floor physical organization. In this case, the shop floor is represented by a grid abstraction, where the shop floor has a chessboard structure with an appearance defined as a matrix of n x m squares.

In order to test the HCA under different configurations and capacities (number of assembly equipment, operation processing capacity per station, assembly equipment per station), an algorithm was developed to generate the input files. The algorithm has as main inputs: **Number of station types**: this input indicates the number of station types that the system can instantiate; **Operation types**: this input is a list of operation types and respective base processing time; λ_{theo}: this parameter represents the maximum number of operation types offered by each station type; λ_{op}: this parameter indicates how many stations can be instantiated at the same time during execution; **Assembly equipment type per station type**. This input includes the average and variance applied for the number of assembly equipment types per station type; **Assembly equipment per station type**. This input includes the average and variance applied for the number of assembly equipment per station. A brief representation of the algorithm can be seen in figure 47. According to the flow-diagram:

1. The system reads the input data, including the main inputs (at the left side of the figure) and the additional inputs (right side of the figure).
2. Based on the number of the station the algorithm generates the station types names, e.g.: STATION_TYPE_1, STATION_TYPE_2;
3. Considering the λ_{theo} the algorithm distributes the operation types per station, attributing unique combinations per station. Additionally, the operation type processing time is changed accordingly to the base time and variance provided as inputs;
4. Using the input "assembly equipment type per station type", the algorithm decides randomly how many assembly equipment types each station requires respecting the average and variance provided. As a rule, independent of how many assembly equipment per station type, the algorithm sets one assembly equipment type in common for all the station types. This extra assembly

equipment type (e.g., rt1) is used later to control how many station types can be instantiated simultaneously.

5. Using the input "assembly equipment per station", the algorithm decides randomly how many assembly equipment each station requires respecting the average and variance provided.
6. The algorithm sets the number of the extra assembly equipment (common to all stations) to λ_{op} this guarantees that the input files will not provide the sufficient conditions to build more stations than λ_{op}.
7. After deciding for the distribution of operations, assembly equipment types per station type, and the total number of assembly equipment, the last step of the algorithm is to create the station_type.json, assembly_equipment_type.json, and assembly_equipment.json files.

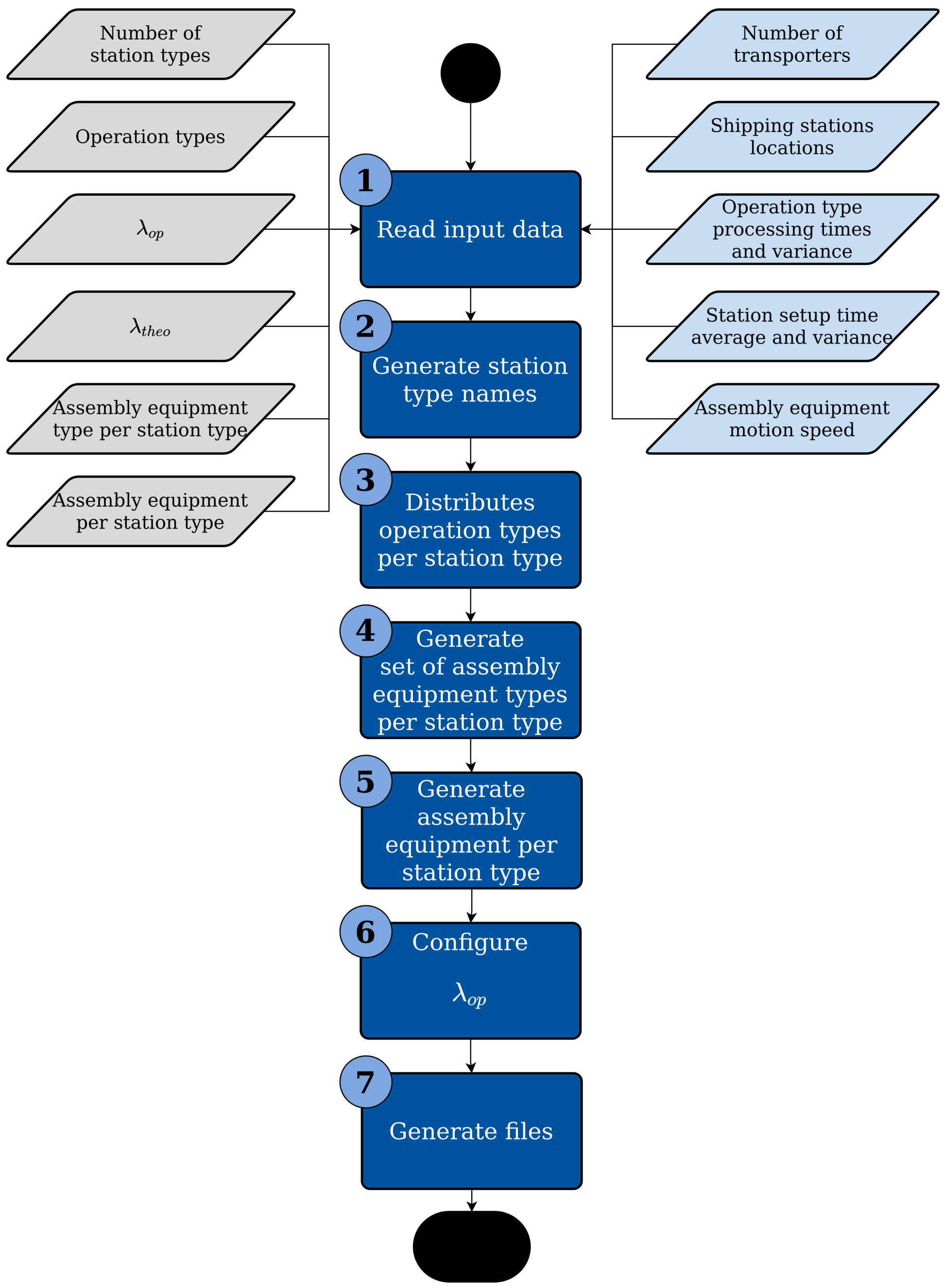

Number of station types
Operation types
λ_{op}
λ_{theo}
Assembly equipment type per station type
Assembly equipment per station type
Number of transporters
Shipping stations locations
Operation type processing times and variance
Station setup time average and variance
Assembly equipment motion speed
1 Read input data
2 Generate station type names
3 Distributes operation types per station type
4 Generate set of assembly equipment types per station type
5 Generate assembly equipment per station type
6 Configure λ_{op}
7 Generate files

Figure 47 – Input file generation diagram.

7 Evaluation

This chapter covers the analysis of the HCA models separated into two parts. The first part shows a detailed analysis of the architecture models considering the coverage of KPIs related to HCA implementation, such as reconfigurability, robustness, maintainability, controllability, complexity, verification, and reusability. The second part executes a quantitative analysis in the HCA prototype, which aims to verify the HCA behavior under the normal (UC1) and disrupted executions (UC2). The drift between the planned and executed during the normal execution is studied under different shop floor and product input configurations. The disrupted scenario is analyzed by the simulation of assembly equipment failure and the response given by two different strategies: LOCAL_FIX and REORGANIZATION.

7.1 Architecture model analysis

In (KRUGER; BASSON, 2018) the authors specify a set of qualitative criteria focussed on the evaluation of holonic control approaches in manufacturing systems: reconfigurability, robustness, maintainability, controllability, complexity, verification, and reusability. This set of requirements is finally associated with a collection of qualitative key performance indicators (KPI): modularity, integrability, diagnosability, convertibility, falt tolerance, distributability, and developer training re1quirents. Table 11 shows a correlation between the KPI mentioned and the LMAS specific requirements:

7.1.1 Modularity

Modularity is a critical aspect and a key enabler for changeable manufacturing systems affecting physical and software components. Modularity in computer science involves the encapsulation and compartmentalization of functionality. According to (BALDWIN; CLARK, 2006), modularity depends on three specifications:

- architecture: identification of software modules;
- interfaces: definition on how the modules interact;
- tests: verification of the behavior of individual and interacting modules.

For the HCA, the software modules are identified in the agents model (c.f. section 5.2.2), where each agent has a unique set of functions, goals, and knowledge that it should provide and rely on, in order to fulfill the system's goals. The agent interaction models (c.f. section 5.2.5) define the external interactions, whose all the agents are responsible

Table 11 – HCA qualitative analysis.

	ID	**Requirements**	**Modularity**	**Integrability**	**Diagnosability**	**Convertibility**	**Fault tolerance**	**Distributability**	**Training requirements**
Resource	RQ-01	Multipurpose station and resources	×	-	-	-	×	-	-
	RQ-02	Fabric model / Shop floor management	×	-	×	-	×	×	×
	RQ-03	Mobile resources / temporary station locations	×	-	-	-	-	-	×
	RQ-04	Spatio-temporal order relations between stations and resources	×	-	-	-	-	×	×
Product	RQ-05	Products, parts and assemblies structures (physical relationships)	×	-	-	×	-	-	-
	RQ-06	Products, parts and assemblies manufacturing data and information	×	-	-	×	-	-	-
Process	RQ-07	Individual order routes	×	-	-	-	-	×	×
	RQ-08	Interface to transportation services	×	-	-	-	-	×	-
Design patterns	RQ-09	Global manufacturing plan	×	-	-	-	-	-	×
	RQ-10	Strategies for delay and failure compensation	×	-	×	-	×	×	×
	RQ-11	Central control / Central monitoring	-	-	×	-	×	×	×
	RQ-12	Decentralized order execution	×	-	×	-	×	×	×
	RQ-13	Ontology-based	×	×	-	×	-	-	×
	RQ-14	Type/Instance	×	-	-	×	-	-	-

for carrying out. The definition contained in the agent model (knowledge, functions, and goals) represents the necessary information that one needs to know in order to create meaningful unit tests regardless of the chosen software implementation language.

7.1.2 Integrability

Integrability is related to a system's ability to quickly and effectively integrate mechanical, informational, and control components with an executing system. This is especially crucial for manufacturing systems that depend on legacy technologies (e.g. not compatible with Industry 4.0 standards). The integrability heavily relies on the programming language chosen and how it facilitates the implementation of new software and technologies. Integrability is then evaluated by the following aspects (KRUGER; BASSON, 2018):

- interfaces to integrate with "foreign" code (e.g., contributions from external partners);
- provision of libraries or functions to implement communication protocols.

According to table 9, the chosen robotic framework is ROS. Officially and counting with the day-to-day community support, ROS supports C++ and Python. However, ROS has "under development" libraries that allow for various other programming languages that are highly used, such as Java and JavaScript. The integrability is even higher when considering the number of libraries and packages available for ROS (e.g., navigation, motion planning, machine vision, drivers for sensor and actuators, 3D models for robotics and equipment). ROS itself is built under the modularity ad integrability concepts. ROS allows developers to re-use code, regardless of the programming language used to write them.

7.1.3 Diagnosability

Diagnosability refers to the system's ease and speed to detect the source of problems and associate a quality. According to (KRUGER; BASSON, 2018) diagnosability is associated with the following factors:

- functionality for construction tests to identify the cause and location of errors;
- the built-in functionality or mechanisms for monitoring communication and execution.

The proposed agent models (c.f. section 5.2.2) provide a "get_status" function for all holon types. The "get_status" function allows external systems to query the current holarchy and get the status of an individual holon or a group of holons. The state model (c.f. section 5.2.4) is part of the holon's status, and monitoring systems can interpret it in order to identify a FAILURE state among the running holons. The holon achieves the FAILURE state due to individual behaviors, but ultimately by a fault (e.g., simulated fault in assembly equipment). Besides the direct assessment of the state model (passive detection strategy), the status contains the executed operations history (e.g., station and order holons operations history), allowing for a specialist system to interpret the current data defining whether the system respects a certain drift tolerance (schedule/executed performance) or if it should find another optimal solution (active detection strategy).

The order, station, and assembly equipment holons contain interactions and internal mechanisms that send the holons to a FAILURE, IN_RESETUP, and IN_REPLAN when a failure is detected in an assembly equipment holon. The holons are equipped with functions to identify these errors, cancel their current operations and assume the respected state.

7.1.4 Convertibility

The convertibility feature is associated with how able the architecture is to use the same functionality to meet new production requirements (e.g., rush order, station shutdown) (KRUGER; BASSON, 2018). In other words, convertibility introduces the idea of changing the manufacturing system while running. The actual architecture models provide a collection of mechanisms that allow the operator and external systems to modify the system's behavior while the system is running. These functions are:

- replan: this functionality is associated with teh station and order holons. The replan aborts the current operation and puts the holon in a state where it is necessary to supply another schedule to make the holon continue its operations;
- resetup: the resetup is associated with the station holon. This service is used to stop the setup process in the station. The station setup process consists of forming the station configuration by setting a collection of assembly equipment. This service is used when an external entity detects a piece of defective equipment during the station setup process;
- pause: the pause operation is associated with the station, order, and sub-assembly holons. The pause puts the holon in a paused state, where operations are not allowed (e.g., assembly operations, move);
- restart: the restart is associated with the station, order, and sub-assembly holons. It puts the holons back to operation;
- update_schedule: this functionality is associated with the order, station, assembly equipment, and part holons. The update schedule can be used to change the initial plan provided by the global optimizer when an external system detects disruptions and decides to change the current schedule;
- load_optimization: this service can be used to load former optimal schedulers. This functionality might be used when the initial state is close to the system state when the results were carried out in terms of resources and customer order. Load optimization facilitates and saves a significant amount of time (e.g., optimization run time).

- update_system_knowledge: this functionality allows an external system or operator to provide new assembly types, part types, assembly equipment type, assembly equipment, shop floor description, and station type to the system. Once received a update_knowledge request, the system updates the type holons, using each type holon's update_knowledge services.

7.1.5 Fault tolerance

Faults will inevitably occur during the execution of the manufacturing system. The causes are diverse: programming errors, machine or controller breakdowns, communication failures, and rush orders. The fault-tolerance refers to remaining operable even when the system faces a fault or a collection of simultaneous or successive faults. The fault tolerance capacity can be characterized by the following aspects(KRUGER; BASSON, 2018):

- fault isolation: it is critical to isolate the errors not to propagate them to the rest of the system;
- fault detection: it is necessary to detect the error and inform the responsible agents quickly;
- fault handling: it is necessary to handle the detected failures accordingly to their nature. This action tries to bring the system back to stability.

The architecture models do not provide a list of specific inputs that could conduct the systems to a state of failure. In other words, this work does not identify types of different failures in its scope. Instead, this work provides a mechanism to deal with failure, the switch mechanism holon (SMH). The SMH monitors failures happening in the station holons. The station holons provide an interface to communicate with assembly equipment holons and listen to possible failures. The failure perception should start at the bottom level (assembly equipment holon). After detection, the failure is transmitted to a higher level holon (station holon). The station holon transmits the information of failure to the SMH that decides whether to order the stations to correct the failure following the current global plan (e.g., replacement of a defective assembly equipment holon) or a complete reconfiguration (save the current operations history, reset the system and require a new global plan considering the defective equipment and the already executed operations). This behavior allows the system to isolate the errors (other stations are not affected by unrelated assembly equipment), identify the error (SMH identifies which station is defective and takes a decision), and handle the actions based on a decision algorithm (this decision algorithm is not part of this dissertation, but the HMS holon is supposed to contain such algorithm).

7.1.6 Distributability

The HMS concepts are related to distributed/decentralized control. The distribution of control enables robustness and portability (KRUGER; BASSON, 2018). When a system execution is distributed among various controllers, it simplifies the application of correction mechanisms, allows for computational resources saving, and improves computational performance. (KRUGER; BASSON, 2018) classify as essential aspects for distributability:

- architecture models should facilitate distribution;
- the communication between distributed control units should be facilitated.

The current architecture distributes the system into six platforms (c.f. table 10): scheduler, database, shop floor, execution control, assembly equipment i, and system knowledge. This platform distribution favors the balance of computational resources. The communication among the distributed control units is facilitated because all the holons should be wrapped into the ROS framework. The ROS framework already gives all the communication protocols for message exchanging and service providing. The legacy or novel system integration should face the same process, ROS wrapping.

7.1.7 Developer training requirements

Developers should understand the HMS concepts to change the system behavior effectively. The availability of professionals aware of the concepts is scarce, and the instruction in HMS and HCA concepts is time-consuming (KRUGER; BASSON, 2018). According to (KRUGER; BASSON, 2018) the developers should be able to:

- implement holon behavior;
- implement concurrency in HCA;
- implement mechanisms for external communication;
- verifies the functionality of the control implementation.

HCA systems are complex and intricate engineering artifacts. The autonomous behavior requires careful analysis and debugging to avoid deadlock conditions and demands a significant amount of time for development. Therefore, the distributed system also requires engineers to understand and deal with complex software techniques for both development and testing.

The developers count on the ANEMONA models (agent, interaction, and organization) to understand the system behavior. The ANEMONA models are composed of individual agent models and their required interactions. The developer understands the agent behaviors through the agent model while the necessary message exchange definition is exposed to the reader with the interaction models. Besides the understating of the HCA models, it is necessary to be trained in the ROS framework. ROS developers count on a broad community of developers and plenty of available tutorials covering basic to advanced concepts. The learning curve for ROS is unavoidably long, but once understood, ROS can serve as an answer for practitioners in the industry, especially when interoperability is a critical requirement.

7.2 Execution analysis

The section evaluates the HCA prototype quantitatively under two situations: UC1 - normal execution; and UC2 - disruption during normal execution. First, the data used as input for customer orders is explained in terms of processing time for a family of pumps. Consecutively, a detailed breakdown of input order complexity and shop floor capabilities is explained (e.g., the maximum number of operations per station, maximum number of simultaneous stations). Ultimately, the UC1 execution is presented, focusing on the delay between the planned and executed schedule scenarios. The tests for UC2 are carried out focusing on the system adaptability performance, considering a simple strategy to recover the system from assembly equipment failures.

7.2.1 Industrial pumps

The customer order inputs consist of multiple pairs of type of pump and its quantity. It is considered pumps of three product families (A, B, and C). The number of operations per pump type is shown in figure 12:

Table 12 – Number of operation in pump families..

Pump type name	Number of operations
PUMP TYPE A1	16
PUMP TYPE A2	16
PUMP TYPE A3	16
PUMP TYPE B1	20
PUMP TYPE B2	20
PUMP TYPE B3	20
PUMP TYPE C1	15
PUMP TYPE C2	15
PUMP TYPE C3	15

Initially, the operations' processing time was known just for PUMP TYPE A1, PUMP TYPE B1, and PUMP TYPE C1. In order to create variability of complexity to challenge the HCA prototype under different conditions, the subsequent pumps of type 2 and 3 for the respective families were generated, multiplying the base time from type 1 by 1.25 and 1.5 respectively (e.g., A1 = 10 s, A2 = 12.5 s, A3= 15 s). Details on the processing times are shown in the appendices A, B, and C.

Originally, the precedence graph of operations was given in the format of a directed graph. The possible sequence of operations is greater than one if one considers all possible variants on how to follow the sequence of the operations. In order to simplify the input, the HCA prototype executes a topological sort algorithm to derive one possible linear sequence of operations. In this way, the operations are executed linearly from the product perspective. This implies that there is no possibility of parallel execution for the same product, but of course, this property does not hold when considering different products. Consequently, when considering the appendices A, B, and C, one should consider the following precedence, figure 48, for the pump families (when creating the types 2 and 3, the precedence is kept).

7.2.2 Input data

In order to generate meaningful and challenging input data (product and shop floor capabilities complexity) for the UC1 and UC2 evaluation cases, two levels of complexity were chosen. The low complexity input data is presented in table 13. The axis Y shows the product complexity, which varies according to the amount products and the blend of product families in the ordered products. The axis X presents the shop floor complexity, which varies according to the number of station types, the maximum number of operations per station, and the maximum number of simultaneous stations.

The high complexity case is presented in table 14. The axes configuration is similar to the low complexity case. The difference is the number of products required on the X-axis. Both tables 13 and 14 presents elements in the form of tuples. The tuples are organized to show the shop floor complexity in the following order:

- number of station types: number of station blueprints offering different capabilities and requirements for their configuration;
- number of maximum operations per station type: the maximum number of operation types offered per station;
- the maximum number of simultaneous station in the shop floor.

An example is $< 8, 2, 3 >$, which means a system that can offer up to eight station types, a maximum of two operations per station type, and a maximum of three stations operating on the shop floor simultaneously. The input data sets were produced utilizing the algorithm described in section 6.5.

Figure 48 – Product families A, B, and C operation precedence order graph.

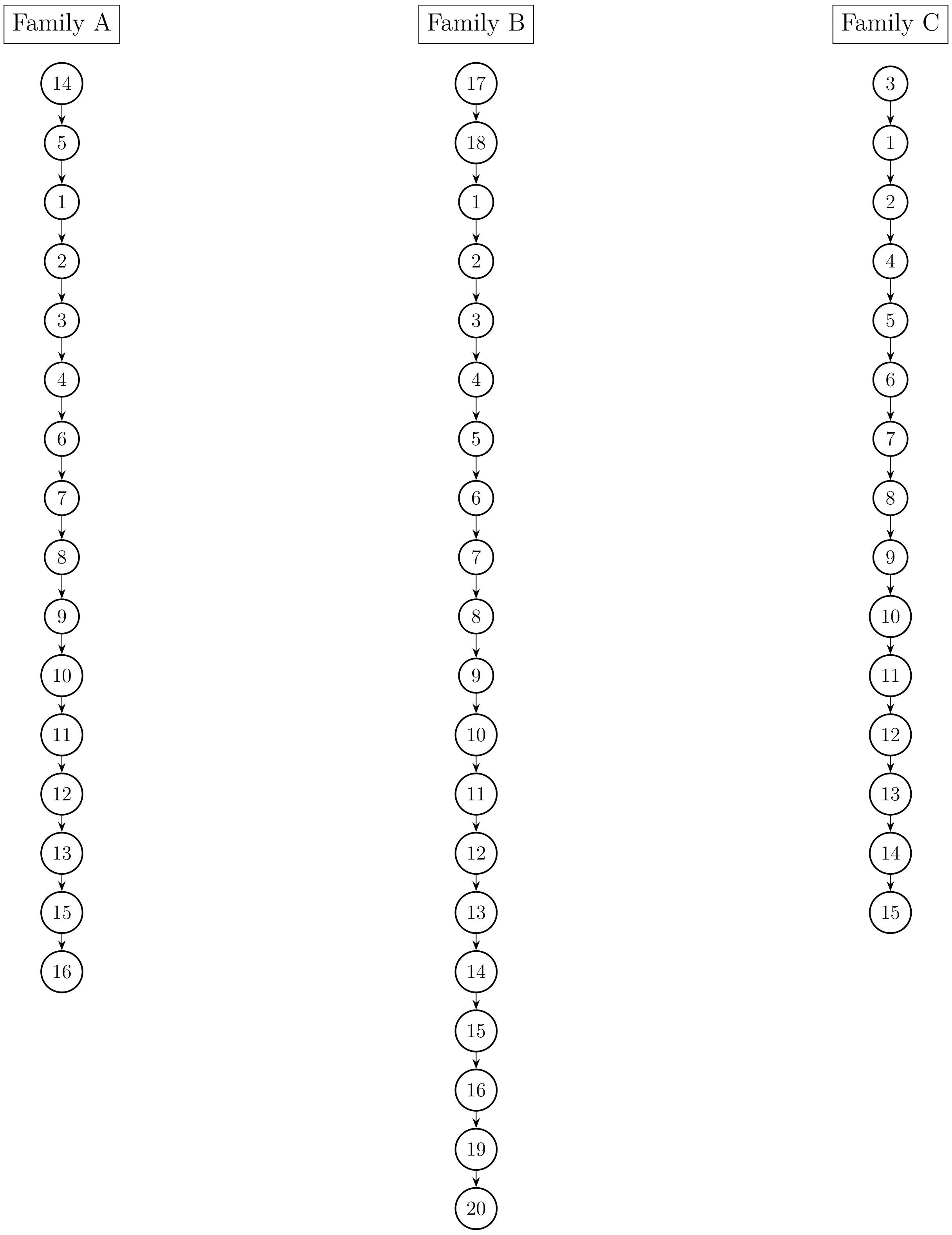


Table 13 – UC1 - Normal use case input: low complexity.

			Manufacturing flexibility		
			low	**medium**	**high**
[2 x PUMP_TYPE_A_1] (32 operations)	**Product variety**	**low**	<16,6,4> id l1	<16,12,6> id l2	<8,12,8> id l3
[1 x PUMP_TYPE_C_1, 1 x PUMP_TYPE_C_2, 1 x PUMP_TYPE_C_3] (45 operations)		**medium**	<8,11,4> id l4	<16,41,6> id l5	<32,41,8> id l6
[1 x PUMP_TYPE_A_3, 1 x PUMP_TYPE_B_2, 1 x PUMP_TYPE_C_1] (51 operations)		**high**	<8,12,4> id l7	<16,47,6> id l8	<32,47,8> id l9

Table 14 – UC1 - Normal use case input: high complexity.

			Manufacturing flexibility		
			low	**medium**	**high**
[4 x PUMP_TYPE_A_1] (64 operations)	**Product variety**	**low**	<16,6,4> id h1	<16,12,6> id h2	<8,12,8> id h3
[1 x PUMP_TYPE_C_1, 2 x PUMP_TYPE_C_2, 2 x PUMP_TYPE_C_3] (75 operations)		**medium**	<8,11,4> id h4	<16,41,6> id h5	<32,41,8> id h6
[2 x PUMP_TYPE_A_3, 3 x PUMP_TYPE_B_2, 1 x PUMP_TYPE_C_1] (107 operations)		**high**	<8,12,4> id h7	-	-

Table 15 – HCA prototype low complexity results.

id	*Gap [%] /Runtime [h]	LMAS completion time [s]	HCA execution time [s]	Drift [s]	Drift rate [%]
l1	80.1/166.0	27311.6	27362.6	51.0	0.2
l2	39.4/166.0	24046.1	24083.1	37.0	0.2
l3	2.7/166.0	24898.1	24986.2	88.1	0.4
l4	0.0/5.6	24770.6	24816.6	46.0	0.2
l5	3.8/166.0	20207.8	20260.9	53.1	0.3
l6	33.5/166.0	19789.0	19880.9	91.9	0.5
l7	0.0/3.5	38301.8	38333.7	31.9	0.1
l8	20.4/166.0	34396.5	34475.8	79.3	0.2
l9	77.5/166.0	33252.2	33332.4	80.2	0.2
				62.0	**Average**

Note: *The gap represents the difference between the current upper and lower bounds (Branch-and-Bound algorithm), with 0 % representing optimality. For cases different than 0 %, The LMAS Scheloc algorithm gap was registered after 166 h of Runtime.

7.2.3 UC1 - Normal execution

7.2.3.1 Execution drift

The execution drift analysis is motivated by the following qualitative question: "How well the HCA prototype follows the LMAS optimization results?". The drift analysis evaluates the differences in execution time of the planned and executed situations for specific input data. The drift analysis consists of the execution of the input data sets presented in section 7.2.2 and the drift analysis followed by reasoning about the possible causes for it.

Table 15 and figure 16 highlight the execution results for the two test batches (low and high complexity). The tables present the drift (LMAS HCA completion time v.s. planned completion time) and the execution time in seconds for both situations. An average delay of 62.0 s is detected in the low complexity execution, and a delay of 108.1 s is detected during the execution of the high complexity test batch. The cases h8 and h9 are not presented because the LMAS optimization algorithm could not generate a suitable schedule within a reasonable time (the optimization was set for a max of 166 h).

Table 16 – HCA prototype high complexity results.

id	*Gap [%] /Runtime [h]	LMAS completion time [s]	HCA execution time [s]	Drift [s]	Drift rate [%]
h1	97.3/166.0	42123.0	42267.8	144.8	0.3
h2	84.2/166.0	26216.4	26290.9	74.5	0.3
h3	15.3/166.0	26358.2	26493.7	135.5	0.5
h4	30.9/166.0	37929.3	38002.3	73.0	0.2
h5	34.7/166.0	21882.0	22001.9	119.9	0.5
h6	62.7/166.0	21545.9	21667.0	121.1	0.6
h7	90.7/166.0	64935.6	65023.5	87.9	0.1
h8	-	-	-	-	-
h9	-	-	-	-	-
				108.1	**Average**

Note: *The gap represents the difference between the current upper and lower bounds (Branch-and-Bound algorithm), with 0 % representing optimality. For cases different than 0 %, The LMAS Scheloc algorithm gap was registered after 166 h of Runtime.

The increase in the drift is expected because of the increase in product complexity. The more the products are under production, it is expected to receive more transport requests and more operation processing requests. Generally, the delay increases as the number of transports increases. Though planned by the global optimizer, each transport and operation requires additional time due to the ROS action server mechanisms. Figure 49 highlights how the drift increases depending on the number of transport requests. However, the drift detected is negligibly relevant compared to the test completion time. The detected drift represents less than 0.6 % (drift rate in tables 15 and 16) of the theoretical completion time in all cases.

As the drift represents a small portion of the theoretical time, it becomes irrelevant compared to the operation execution time order of magnitude. For this reason, the theoretical throughput derived from the LMAS optimization algorithm is not changed in a significant way. The drift analysis is essential because it shows how the designed models, in the form of an implemented prototype, performed regarding the reliability according to the theoretical optimal schedule. Classical analysis such as throughput or resource usage analyzes the optimization algorithm but not the HCA.

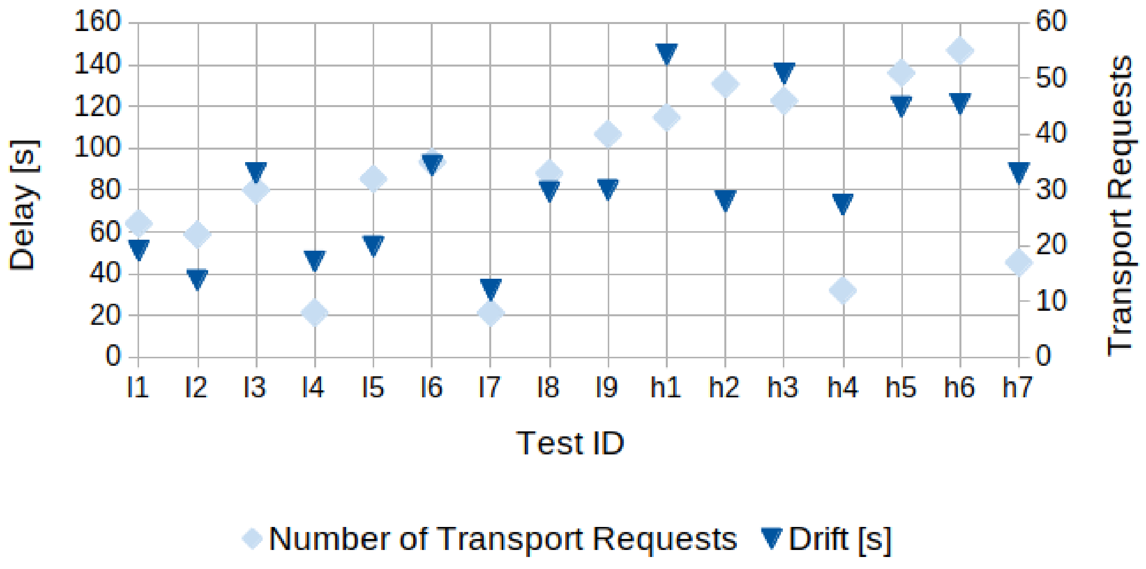


Figure 49 – HCA prototype execution drift analysis.

7.2.4 UC2 - Disruption

7.2.5 Test preparation

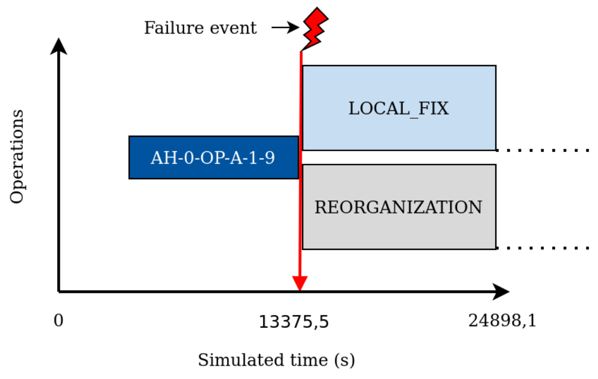
The following section evaluates the deployed disruption model described in subsection 5.2.5.5: the local fix: replacement of a defective assembly equipment holon; and reorganization: complete reoptimization of the system in the point of failure.

The prototype was implemented to deal with a very simple disruption: a failure is triggered in the lowest level (assembly equipment holon) containing the strategy to be followed by the switch mechanism holon, local fix (heterarchical), or complete reorganization (hierarchical). After receiving the failure simulation request, the assembly equipment holon follows the procedure described in section 5.2.5.5. In order to test the disruption model, two data sets were selected (id l3 of table 13 and id h4 of table 14). The procedure adopted to test is the following:

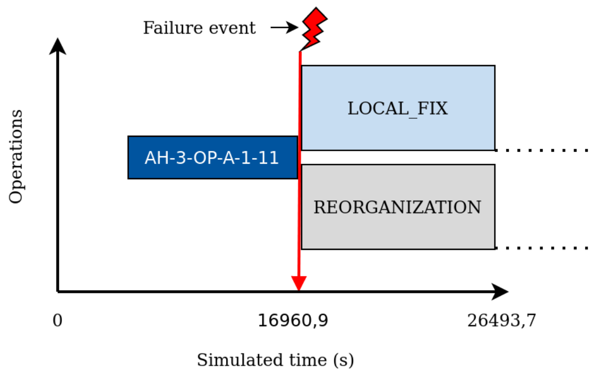
1. Select an operation in which a failure will be simulated to test the disruption model;
2. Execute the architecture prototype;
3. Wait for point in the simulated time, when the selected operation finishes, to simulate the failure containing an indication to follow a local fix or a complete reorganization;
4. Simulate the failure in an assembly equipment holon associated with the station holon in which the targeted operation is being executed;
5. Wait until the execution is complete;

6. Analyze the results;

The objective is to test the two approaches and evaluate which one produces a final solution with the smallest completion time value. A visual representation eases the comprehension of the temporal aspect of the input data set in figure 50a and 50b. Adopting the described procedure for the two input test data sets, and for the two disruption handling strategies, a test recipe is produced, and it is represented in the table 17. The two strategies are triggered at the end of a selected operation for each test.



(a) Id l3 UC2 test representation.



(b) Id h3 UC2 test representation.

Table 17 – UC2 test recipe.

#	id	Operation	Strategy	AEH	SH	Expected failure time [s]
1	l3	AH-0-OP-A-1-9	LOCAL_FIX	rt1_2_13	SH_5	13375.5
2	l3	AH-0-OP-A-1-9	REORGANIZATION	rt1_2_13	SH_5	13375.5
3	h3	AH-3-OP-A-1-11	LOCAL_FIX	rt2_9_20	SH_3	16960.9
4	h3	AH-3-OP-A-1-11	REORGANIZATION	rt2_9_20	SH_3	16960.9

# -> test trial sequence number | AEH -> Assembly equipment holon | SH -> Station holon

7.2.6 Test results

The test execution was performed in the same configuration as the normal case in section 7.2.3 (e.g. simulated environment). An example of execution is exhibited for the rounds #1 (local fix) and #2 (reorganization) in figures 51 and 52 respectively. Figures 53 and 54 show a scaled-down version of the completion graphs for the id l3 and h3 respectively. These figures enable the reader to identify the time consumed during the local fix and the optimization. Here, just the period not associated with the handling is scaled down. The final results, also including the cases #3 and #4, are compiled in the table 18.

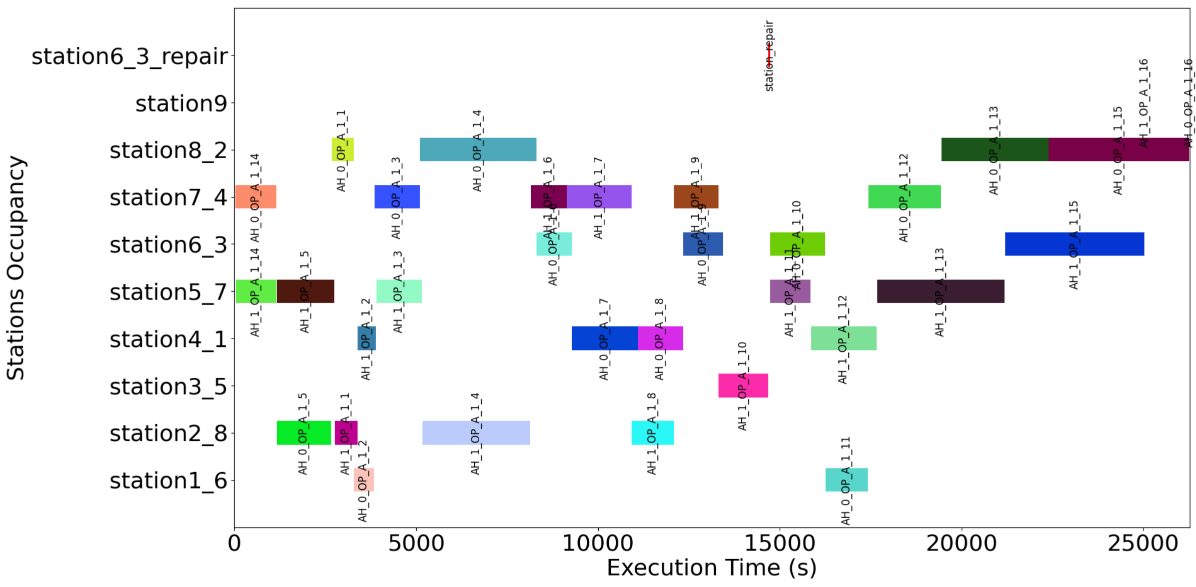


Figure 51 – Id l3 local fix completion graph.

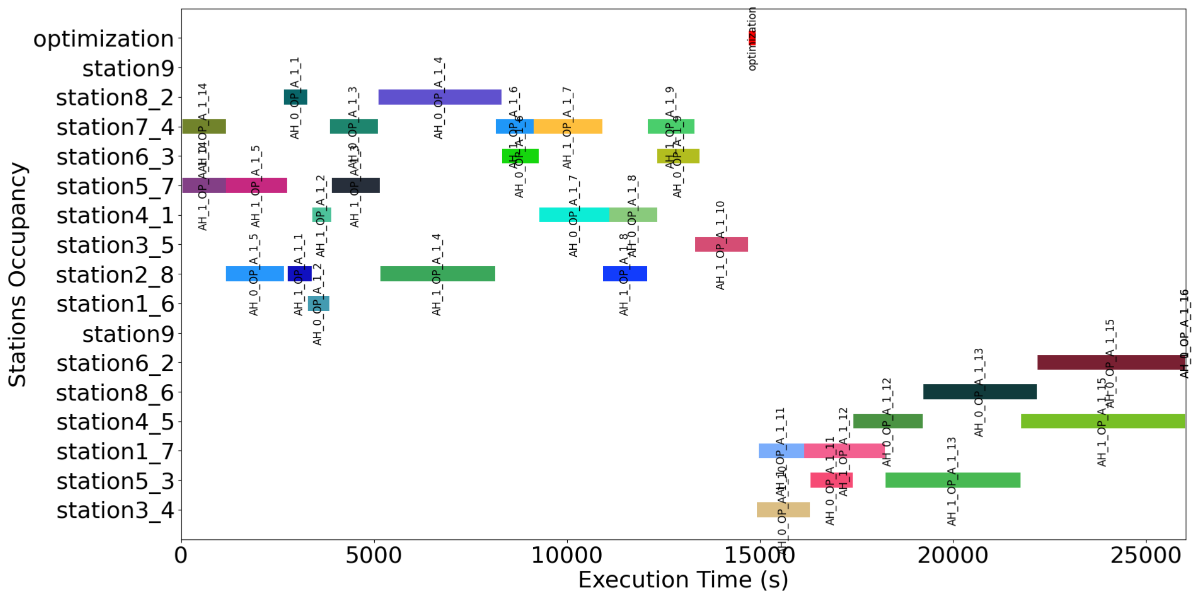


Figure 52 – Id l3 reorganization completion graph.

7.2.7 Test analysis

The outcome from the execution provided results for further analysis.

- Disruption handling executed on a different point in time: depending on the execution state and the number of affected order holons, the system might not stop right after the completion of the selected operation for disruption. This behavior is observable because the architecture prototype was implemented

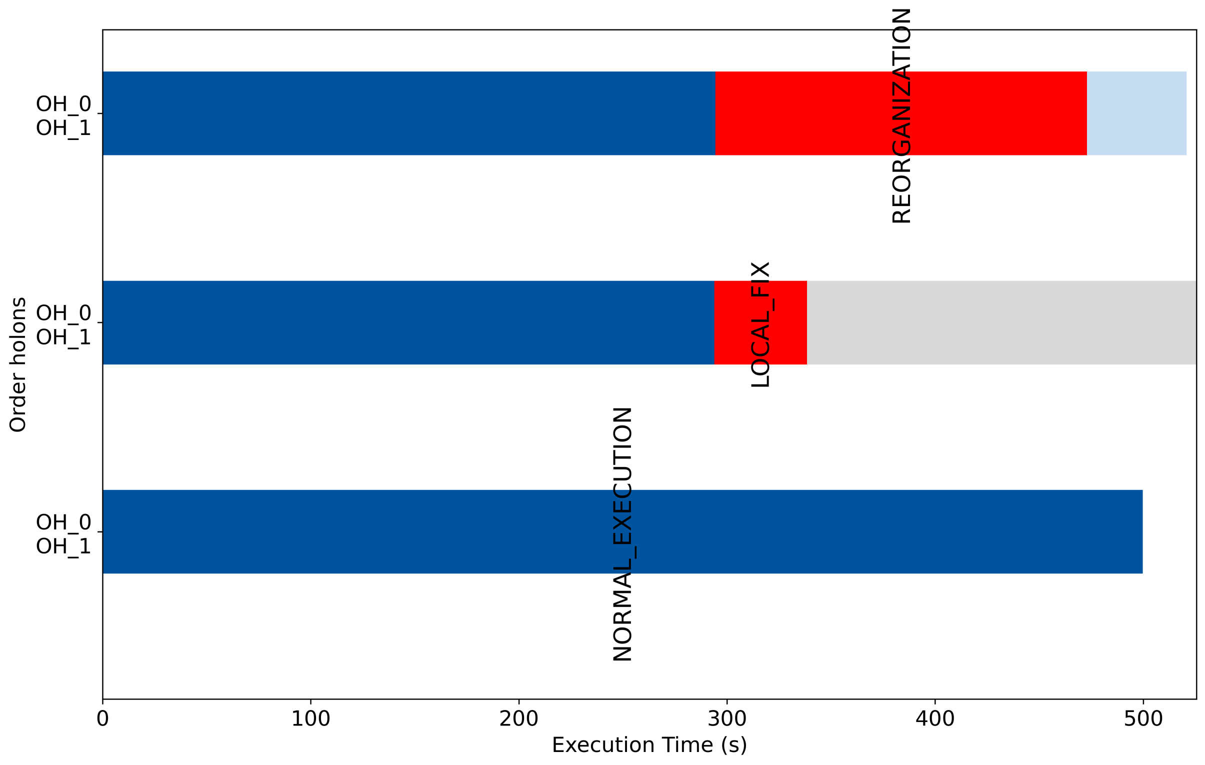


Figure 53 – Id l3: Local fix vs. Reorganization strategies scaled down completion graph. LOCAL_FIX and REORGANIZATION are scaled to 1. The remaining time is scaled down 50 x.

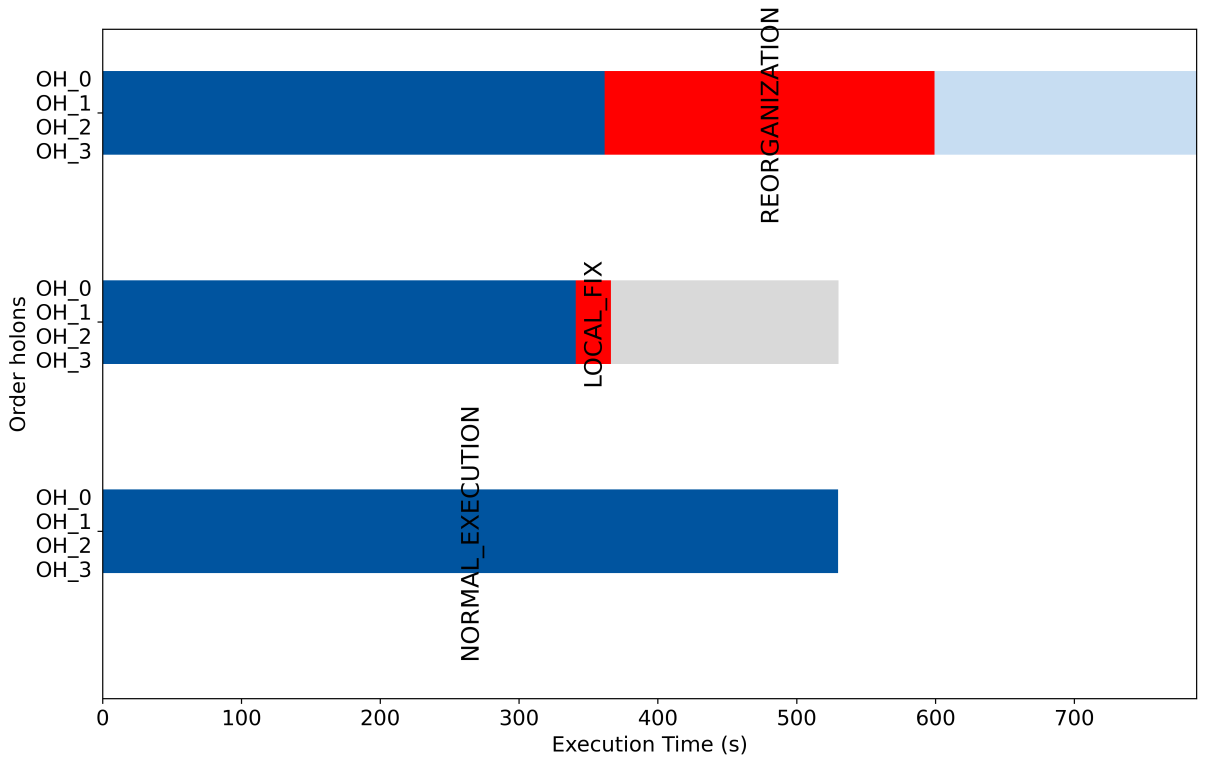


Figure 54 – Id h3: Local fix vs. Reorganization strategies scaled down completion graph. LOCAL_FIX is scaled to 1. The remaining time is scaled down 50 x.

Table 18 – UC2 execution results.

Step	**l3**	**h3**
Normal execution completion time [s]	24986.2	26493.7
Local fix completion time [s]	26279.3	26505.6
Reorganization completion time [s]	26037.8	39405.9
Local fix duration [s]	44.6	25.4
Reorganization duration [s]	178.5	11889.2
Failure time local fix [s]	14690.1	17039.4
Failure time reorganization [s]	14716.7	18076.8
Optimization gap [%]	0.0	39.7

considering that each affected order holon should be stopped in sequence. In order to stop each order holon, the order holon should not be executing any operation. If an operation is being executed, the order holon waits until its completion. As the order holons are stopped sequentially, order holons that were not informed continue their execution, and that can cause delays in the disruption handling;

- LOCAL_FIX run-time shorter than the REORGANIZATION run-time: The LMAS optimization algorithm has a complexity that scales up quickly, depending on the number of operations. That means, in cases where the number of operations is high, the optimization algorithm might require a long run time to generate a feasible solution. That general hypothesis promotes the idea that the local fix is always a quick solution when it comes to completion time. However, for some instances, when the number of operations to be executed after the failure is not significant, the optimization algorithm produces results comparable with the local fix time, which in specific situations can generate a reduced value of completion time compared with the local fix solution. That case is observed for id l3 in the figure 53.

8 Conclusion

This dissertation executed the action research methodology to answer the research question "How does an LMAS production and control system architecture need to be designed?". The introductory section presented the justification for an HCA approach for LMAS based on the increased flexibility and customization carried by the Industry 4.0 context and the related requirements mandatory for an LMAS HCA application. Ground concepts such as HMS, agents, Industry 4.0, and Cyber Production System were presented in chapter 3.

Chapter 4 revised relevant contributions in the field of HMS/HCA, and research works in the area of manufacturing ontologies that have a significant impact on the LMAS paradigm. The revision over the related HCA scientific reference contributions identified the need to define a novel HCA architecture respecting the requirements of LMAS. The definition of an HCA for LMAS was primarily motivated by the lack of interaction model definitions for individual work routes and explicit dynamic station configuration features. Additionally, during the revision of reference manufacturing ontologies, the need to define an ontology covering the main concepts used in the HMS and LMAS supporting the agent and interaction models of ANEMONA models was identified. The definition of a new ontology, namely Holonotlogy, in a .owl format was motivated by the lack of public-available contributions in the digital format of ontologies that hold concepts related to holonic systems. The specific objective of "HMS methodology definition" was completed in this chapter.

In the first part of chapter 5, the list of specific LMAS requirements, simplifications, and goals was summarized. The LMAS requirements were inspired by the UC1 and UC2 presented in chapter 3. The analysis of both user case diagrams originated a list of 63 requirements that need to be respected by the architecture models. The list of 25 simplifications summarizes the relaxation applied in the models and consequently is reflected in the prototype. It was elected 11 goals for the LMAS systems based on previous contributions in the LMAs universe and specific aspects of this dissertation such as "Launch and control the system dynamic station setup process utilizing the assembly equipment available on the shop floor".

The second part of chapter 5 presented the core contribution of this dissertation: the architecture models (organization, agent, agent state, interaction, and environment models) that conceptualize the HCA approach for LMAS. The models were designed following the ANEMONA methodology, a multi-agent methodology specially designed for holonic manufacturing systems. Unlike the revised related work, the interaction and agent models covering the station holon were especially considered an individual holon and an abstract agent following the ANEMONA methodology (agent formed by the collaboration of other agents). The interaction models considered the interaction units necessary to unify temporarily a set of agents and operations to dissolve them when required in other places or processes. The Order execution interaction diagram covered the individual order routes, where order, assembly equipment, sub-assembly, station, part, and database holons collaborate to guide the parts and control the operation execution through the shop floor. As a complement for the agent models, the state models for the main holons (assembly equipment, station, part, station, and sub-assembly holons) are presented. The state models contain the states and transition functions necessary to translate the agent behavior specified by the models to the holons implementation serving simultaneously as a basis for entities interested in the actual state of external entities. The specific objective of "design of HMS architecture models" was completed in this chapter.

The Disruption handling interaction diagram presented the model that captures the behavior executed when dealing with failures. The declared behaviors (LOCAL_FIX and REORGANIZATION) are a simple correction mechanism, but their implementation emphasizes the need to deal with disruptions dynamically and serves as a basis to test cases for UC2. LOCAL_FIX deals with a simple replacement of the defective assembly equipment while the REORGANIZATION deals with the system total reconfiguration using the optimization layer to optimize the considering operations to be executed from the point where the failure was detected.

Chapter 6 key finding was the implementation aspects used in the HCA prototype, where the author uses the ROS framework to implement the models. Chapter 6 presented the result of an essential step of the ANEMONA methodology, the agent template, which the author adapted to create the ROS Template containing ROS-based terminology such: services, topics, and action servers. Additionally, in chapter 6, the development framework to create an ontology called Holontology was presented. Holontology is composed of 2 sub ontologies, also created in the context of this work, that contain concepts of cyber and physical worlds, respectively. The main elements of Holontology were inspired by the MASON and MaRCO reference manufacturing ontologies. A data model was created based on Holonology and applied to the HCA prototype. This data model can be considered as an extension to the ANEMONA models. In order to test the prototype with challenging input data, an algorithm for an input data test generator was presented. It utilizes sewage pump assembly information from industry and input about additional randomness on operation execution times, the number of maximum different operations per station (capabilities), as well as the average count of resources per station and its variance. Towards the software implementation, chapter 6 presents the reasoning for using ROS as the standard framework for the prototype and the ROS meta-packages used in the implementation. The following specific objectives were completed in this chapter: "design and implementation of a data model based on a higher-level ontology"; "implementation of the HMS architecture utilizing a robotic software framework platform as a prototype"; "integration of the architecture prototype with former modules of LMAS".

Chapter 7 showed a detailed analysis of the proposed architectural models. The analysis was two-folded. Therefore, the first part focuses on evaluating the qualitative aspects: integrability, diagnosability, convertibility, fault tolerance, distributability, and developer training requirements. The second part is focused on the time difference between the global schedule and the executed scenario in the normal execution (UC1) and during disruption handling (UC2). A procedure was presented to test the normal use case (without disruptions) under low and high complexity cases, highlighting the scheduled and executed scenario differences. The difference proved irrelevant or insignificant, on average less than 0.6 % of the completion time, due to the nature of the operation execution times (long execution). The delays are related to the communication mechanism applied for the prototype use: ROS action and normal servers. For the UC2, one low and high complexity cases were selected in order to verify the behavior proposed by the models for two different strategies (LOCAL_FIX and REORGANIZATION), followed by an analysis that indicates when one or the other scenario might be beneficial to choose in order to handle the disruption and finish the execution in the least time as possible. In a general evaluation, the LOCAL_FIX should be used when there are not a considerable number of operations to be executed after the moment of failure. In that case, the REORGANIZATION would require a significant runtime to optimize the new schedule. The following specific objectives were completed in this chapter: "evaluation of the prototype's software logic and proposed holonic features in a simulated environment"; "quantitative and qualitative analysis of the simulated trials in order to support the answer to be given to the research question". In summary, the author elects the following topics as main guidelines to answer the research question:

- Architecture models: the architecture should follow a multi-agent methodology in order to create the conceptual models. Multi-agent models bring best practices and guidelines to design optimal distributed software systems. The advantages of the multi-agent systems are related to the distributed processing of operations, enabling the system to act quickly on income failure even to collaborate to follow an optimal global plan given by one specialized entity. The conceptual models should be aligned with the requirements for an LMAS, converting the hierarchical and heterarchical demands. The author particularly recommends ANEMONA methodology due to the bias to the holonic manufacturing systems; a concept developed aiming to benefit from the best of hierarchical and heterarchical worlds. The models are mandatory and act as an architecture recipe structuring how the holons in the system should behave depending on upcoming internal and external events. The individual order routes and station reconfigurability should be presented in the models to respect the LMAS essential dynamic requirements;

- Ontology: the ANEMONA's Agent Template (modified in this work to adapt to the ROS reality) requires the definition of an ontology for the agent data model. Inspired by this requirement and by the latest development in the semantic web and its advantages (e.g., interoperability), the author strongly recommends starting the development of an HCA from an ontology. Following that recommendation, the researcher will soon be challenged with the definition of interaction and agent data concepts that later on can be added to the interaction and agent models, regardless of the chosen methodology. Ontology is a technology that has gained a revival of emphasis in the last years in the manufacturing field. The revival might be related to the interoperability requirements imposed by the complex systems that compose a typical CPPS;
- Robot Operating System (ROS): The ANEMONA methodology presents the JAVA framework library JADE as the standard to enable the agent behavior. However, the author, inspired by his background and supported by the latest developments in robotics, suggests applying a different framework (ROS) when the intention, later on, is to apply the architecture to highly automated and robotized systems. ROS presents numerous advantages when it comes to the re-use of specialized algorithm code packages (e.g., mapping, navigation, SLAM, dynamic motion planning) and "do not invent the wheel if you already have the wheels to use";
- Disruption handling: To enable the system to undergo disruption scenarios robustly (e.g., defective components, rush orders, and misalignment between executed and scheduled scenarios), the author highly recommends considering in the interaction models strategies to deal with such scenarios. The strategies might use the entity's autonomy or follow a more rigid procedure. That depends on a reasoning mechanism and the system's stability (a chaotic state might result from a highly competitive agent system). The strategies applied were the LOCAL_FIX and the REORGANIZATION. When the objective is to reduce the completion time, the preference for choosing between them varies according to the execution state. If the system is interrupted "close enough" to the current job input termination, the required optimization run time is not significant, promoting the REORGANIZATION strategy. When there are a significant amount of operations to allocate, the preferred strategy becomes the LOCAL_FIX.;
- Simulated prototype: the author recommends investing time for a prototype

realization that could later be used as an evaluation mechanism alongside the real system. The prototype allows for detecting implementation errors and model limitations. It is an essential tool for an agile methodology practice when the client needs to evaluate small but meaningful work packages during the development cycle.

Although beneficial in various ways (e.g., integration, publisher/subscriber abstraction, reusability of code), the use of ROS for multi-agent systems brings an additional load of work when it is necessary to model and implement high-level agent behavior; ROS is not an agent specific platform. ROS could be used with multi-agent specific platforms as an alternative solution in future contributions, bringing the agent abstraction and interoperability together. The use of the ontological approach presents numerous advantages, as cited before. Although intended for a higher level of operations, such as inferring data in a specific domain, this work's result ontology serves mainly to present a structured manner to derive a data set from it

The ANEMONA methodology brings a set of guidelines and the necessary documentation to understand how to apply multi-agent systems in HCA models. As a consequence of its extensive documentation, the execution of the method to apply ANEMONA is lengthy due to the high number of steps, slow learning curve, and iterations necessary for model refinement. Also, the availability of examples with variations of the provided notation is scarce (excluding those in the textbook). Making use of such methodology makes one rely mainly on the authors' main textbook. The unavailability of an ANEMONA design tool also requires the development of drawings and figures by its users, which might lead to misunderstandings in different solution comparisions.

This dissertation does not focus on an intricate mechanism to decide which approach to follow during a disruption scenario. That could be understood as a reasoning entity that autonomously decides what strategy to follow (e.g., targeting a specific objective: least completion time, save energy, save resources) instead of requiring user input. The automatic translation from an ontology artifact (e.g., .owl file) to the data model applied in the prototype is not covered. The architecture prototype was not tested in a distributed system. The drift between executed and scheduled scenarios might be higher in such a case.

As future contributions, the results presented in this work suggest the need for more powerful disruption handling of operations occurring in industrial practice (e.g., allocation of operations based on alternatives in their operation sequences); the comparison of the architecture execution with a DES simulated environment approach, which is a tool that might be useful when it comes to the simulation of execution uncertainties and unpredicted behaviors; the decision-making among available disruption handling strategies is a problem to be investigated; in other words, how to decide to follow a specific strategy instead of others based on preset manufacturing objectives (e.g., throughout, completion time, transport time); and the behavior of the architecture prototype when connected to actual equipment, which would impose an increase in delays and robustness to deal with uncertainties of the actual assembly process.

Bibliography

AHELEROFF, S.; ZHONG, R. Y.; XU, X. A Digital Twin Reference for Mass Personalization in Industry 4.0. **Procedia CIRP**, v. 93, p. 228–233, 2020. ISSN 22128271.

AMERI, F.; DUTTA, D. An upper ontology for manufacturing service description. In: ASME (Ed.). **Proceedings of the 2006 ASME International Design Engineering Technical Conferences & Computers and Information In Engineering Conference, September 10-13, 2006**. New York: ASME, 2006. ISBN 079183784X.

BALDWIN, C. Y.; CLARK, K. B. Modularity in the Design of Complex Engineering Systems. In: . [S.l.: s.n.], 2006.

BARBOSA, J.; LEITÃO, P.; ADAM, E.; TRENTESAUX, D. Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution. **Computers in Industry**, v. 66, p. 99–111, 2015. ISSN 01663615.

BASYS. Eclipse BaSyx. feb 2021. Available at: <https://projects.eclipse.org/projects/technology.basyx>.

BELLIFEMINE, F.; CAIRE, G.; POGGI, A.; RIMASSA, G. JADE: A software framework for developing multi-agent applications. Lessons learned. **Information and Software Technology**, v. 50, n. 1, p. 10–21, 2008. ISSN 0950-5849. Special issue with two special sections. Section 1: Most-cited software engineering articles in 2001. Section 2: Requirement engineering: Foundation for software quality. Available at: <https://www.sciencedirect.com/science/article/pii/S0950584907001218>.

BOISSIER, O.; BORDINI, R. H.; HÜBNER, J.; RICCI, A. **Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo**. MIT Press, 2020. ISBN 9780262044578. Available at: <https://mitpress.mit.edu/books/multi-agent-oriented-programming>.

BORANGIU, T.; RĂILEANU, S.; BERGER, T.; TRENTESAUX, D. Switching mode control strategy in manufacturing execution systems. **International Journal of Production Research**, v. 53, n. 7, p. 1950–1963, 2015. ISSN 0020-7543.

BORGO, S.; LEITÃO, P. The Role of Foundational Ontologies in Manufacturing Domain Applications. **On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE**, v. 3290, p. 670–688, 2004.

BOTTI, V.; GIRET, A. **ANEMONA: A Multi-Agent Methodology for Holonic Manufacturing Systems**. 1st. ed. London: Springer-Verlag London, 2008. ISBN 1848003099.

BRUSSEL, H.; WYNS, J.; VALCKENAERS, P.; BONGAERTS, L.; PEETERS, P. Reference architecture for holonic manufacturing systems: PROSA. **Computers in Industry**, v. 37, p. 255–274, 11 1998.

BUCKHORST, A. F.; GRAHN, L.; SCHMITT, R. H. Decentralized Holonic Control System Model for Line-less Mobile Assembly Systems (LMAS). **Journal Name: Robotics and Computer-Integrated Manufacturing**, 2021, in Press.

BUCKHORST, A. F.; HÜETTEMANN, G.; GRAHN, L.; SCHMITT, R. H. Assignment, Sequencing and Location Planning in Line-less Mobile Assembly Systems. In: SCHÜPP-STUHL, T.; TRACHT, K.; ROSSMANN, J. (Ed.). **Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019. p. 227–238. ISBN 978-3-662-59317-2.

BUCKHORST, A. F.; MONTAVON, B.; WOLFSCHLÄGER, D.; BUCHSBAUM, M.; SHAHIDI, A.; PETRUCK, H.; KUNZE, I.; PENNEKAMP, J.; BRECHER, C.; HÜSING, M.; CORVES, B.; NITSCH, V.; WEHRLE, K.; SCHMITT, R. H. Holarchy for Line-less Mobile Assembly Systems Operation in the Context of the Internet of Production. In: ELSEVIER (Ed.). **Proceedings of the 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering (ICME '20), July 14-17, 2020**. Gulf of Naples, Italy: Elsevier, 2020. Available at: <https://jpennekamp.de/wp-content/papercite-data/pdf/bmw+20.pdf>.

BUCKHORST, A. F.; SCHMITT, R. H. Multi-Staged, Multi-Objective Optimization for Operation Management in Line-less Mobile Assembly Systems (LMAS). **Procedia CIRP**, v. 93, p. 1121–1126, 2020. ISSN 2212-8271. 53rd CIRP Conference on Manufacturing Systems 2020. Available at: <https://www.sciencedirect.com/science/article/pii/S2212827120306119>.

BUSSMANN, S.; JENNINGS, N. R.; WOOLDRIDGE, M. **Multiagent Systems for Manufacturing Control: A Design Methodology**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. (Springer Series on Agent Technology). ISBN 9783662088722.

CHIRN, J.-L.; MCFARLANE, D. C. A holonic component-based approach to reconfigurable manufacturing control architecture. In: IEEE (Ed.). **Proceedings of the 11th International Workshop on Database and Expert Systems Applications, 4-8 Sept. 2000**. London, UK: IEEE, 2000. p. 219–223.

CHRISTENSEN, J. Holonic Manufacturing Systems Initial Architecture and Standards Directions. In: **Proceedings of the First European Conference on Holonic Manufacturing Systems, 1 December, 1994**. Hannover, Germany: First European Conference on Holonic Manufacturing Systems, 1994.

DERIGENT, W.; CARDIN, O.; TRENTESAUX, D. Industry 4 contributions of holonic manufacturing control architectures and future challenges. **Journal of Intelligent Manufacturing**, 2020. ISSN 0956-5515.

FRANKLIN, S.; GRAESSER, A. Is It an agent, or just a program?: A taxonomy for autonomous agents. In: CARBONELL, J. G.; SIEKMANN, J.; GOOS, G.; HARTMANIS, J.; van Leeuwen, J.; MÜLLER, J. P.; WOOLDRIDGE, M. J.; JENNINGS, N. R. (Ed.). **Intelligent Agents III Agent Theories, Architectures, and Languages**. Berlin, Heidelberg: Springer Nature, 1997, (Lecture Notes in Computer Science, v. 1193). p. 21–35. ISBN 978-3-540-62507-0.

GARCIA, E.; GIRET, A.; BOTTI, V. **Regulated open multi-agent systems**. New York: Springer, 2014. ISBN 9783319115719.

GAUSEMEIER, J.; MOEHRINGER, S. VDI 2206- A New Guideline for the Design of Mechatronic Systems. **IFAC Proceedings Volumes**, v. 35, n. 2, p. 785–790, 2002. ISSN 1474-6670. 2nd IFAC Conference on Mechatronic Systems, Berkeley, CA, USA, 9-11 December.

GIRET, A.; BOTTI, V. Holons and agents. **Journal of Intelligent Manufacturing**, v. 15, n. 5, p. 645–659, 2004. ISSN 0956-5515.

GIRET, A.; TRENTESAUX, D.; SALIDO, M. A.; GARCIA, E.; ADAM, E. A holonic multi-agent methodology to design sustainable intelligent manufacturing control systems. **Journal of Cleaner Production**, v. 167, p. 1370–1386, 2017. ISSN 09596526.

GOU, L.; LUH, P. B.; KYOYA, Y. Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation. **Computers in Industry**, v. 37, n. 3, p. 213–231, 1998. ISSN 0166-3615.

GRÄSSLER, I.; PÖHLER, A. Implementation of an Adapted Holonic Production Architecture. **Procedia CIRP**, v. 63, p. 138–143, 2017. ISSN 22128271.

GUARINO, N. Ontologies and knowledge bases: towards a terminological clarification. **Towards Very Large Knowledge Bases**, p. 25–32, 1995.

HERMANN, M.; PENTEK, T.; OTTO, B. Design principles for industrie 4.0 scenarios. In: IEEE (Ed.). **2016 49th Hawaii International Conference on System Sciences (HICSS)**. Koloa, HI, USA: IEEE, 2016. p. 3928–3937.

HÜETTEMANN, G.; BUCKHORST, A.; SCHMITT, R. Modelling and Assessing Line-less Mobile Assembly Systems. **Procedia CIRP**, v. 81, p. 724–729, 01 2019.

IÑIGO-BLASCO, P.; RIO, F. Diaz-del; ROMERO-TERNERO, M. C.; CAGIGAS-MUÑIZ, D.; VICENTE-DIAZ, S. Robotics software frameworks for multi-agent robotic systems development. **Robotics and Autonomous Systems**, v. 60, n. 6, p. 803–821, 2012. ISSN 09218890.

JÄRVENPÄÄ, E.; SILTALA, N.; HYLLI, O.; LANZ, M. The development of an ontology for describing the capabilities of manufacturing resources. **Journal of Intelligent Manufacturing**, v. 30, n. 2, p. 959–978, 2019. ISSN 0956-5515.

JIMENEZ, J.-F.; BEKRAR, A.; ZAMBRANO-REY, G.; TRENTESAUX, D.; LEITÃO, P. Pollux: a dynamic hybrid control architecture for flexible job shop systems. **International Journal of Production Research**, v. 55, n. 15, p. 4229–4247, 2017. ISSN 0020-7543.

KOESTLER, A. **The Ghost in the Machine**. New York: Arkana Books, 1968.

KRUGER, K.; BASSON, A. Evaluation criteria for holonic control implementations in manufacturing systems. v. 32, p. 1–11, 11 2018.

LAMNABHI-LAGARRIGUE, F.; ANNASWAMY, A.; ENGELL, S.; ISAKSSON, A.; KHARGONEKAR, P.; MURRAY, R. M.; NIJMEIJER, H.; SAMAD, T.; TILBURY, D.; Van den Hof, P. Systems & Control for the future of humanity, research agenda: Current and future roles, impact and grand challenges. **Annual Reviews in Control**, v. 43, p. 1–64, 2017. ISSN 1367-5788. Available at: <https://www.sciencedirect.com/science/article/pii/S1367578817300573>.

LEE, E. A.; SESHIA, S. A. **Introduction to embedded systems: A cyber-physical systems approach**. Second edition. Cambridge, Massachuetts: The MIT Press, 2017. ISBN 9780262533812.

LEE, J.; BAGHERI, B.; KAO, H.-A. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. **Manufacturing Letters**, v. 3, p. 18–23, 2015. ISSN 22138463.

LEGAT, C.; SEITZ, C.; LAMPARTER, S.; FELDMANN, S. Semantics to the Shop Floor: Towards Ontology Modularization and Reuse in the Automation Domain. **IFAC Proceedings Volumes**, v. 47, n. 3, p. 3444–3449, 2014. ISSN 1474-6670. 19th IFAC World Congress.

LEITÃO, P.; RESTIVO, F. ADACOR: A holonic architecture for agile and adaptive manufacturing control. **Computers in Industry**, v. 57, n. 2, p. 121–130, 2006. ISSN 01663615.

LEMAIGNAN, S.; SIADAT, A.; DANTAN, J.-Y.; SEMENENKO, A. MASON: A Proposal For An Ontology Of Manufacturing Domain. In: IEEE (Ed.). **Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. 15-16 June, 2006**. Prague, Czech Republic: IEEE, 2006. p. 195–200. ISBN 0-7695-2589-X.

MONOSTORI, L.; KÁDÁR, B.; BAUERNHANSL, T.; KONDOH, S.; KUMARA, S.; REINHART, G.; SAUER, O.; SCHUH, G.; SIHN, W.; UEDA, K. Cyber-physical systems in manufacturing. **CIRP Annals**, v. 65, n. 2, p. 621–641, 2016. ISSN 00078506. Available at: <https://www.sciencedirect.com/science/article/pii/S0007850616301974>.

NIST. **Foundations for innovation: Strategic R&D opportunities for 21st century cyber-physical systems: Connecting computer and information systems with the physical world. Report of the Steering Committee for Foundations in Innovation for cyber-physical systems**. US: NIST, 2013.

NWANA, H. S. Software agents: an overview. **The Knowledge Engineering Review**, v. 11, n. 3, p. 205–244, 1996. ISSN 0269-8889.

OBITKO, M.; VRBA, P.; MAŘÍK, V. Applications of Semantics in Agent-Based Manufacturing Systems. **Informatica (Informatica-Lithuan)**, v. 34, p. 315–330, 2010.

OLIVEIRA, J. A. B. de. **Towards SMART Manufacturing**. Lisbon, 2019.

PACH, C.; BERGER, T.; BONTE, T.; TRENTESAUX, D. ORCA-FMS: a dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling. **Computers in Industry**, v. 65, n. 4, p. 706–720, 2014. ISSN 01663615.

PAL, C.-V.; LEON, F.; PAPRZYCKI, M.; GANZHA, M. A Review of Platforms for the Development of Agent Systems. **Computer Science (ArXiv)**, 2020.

PARK, K.-J.; ZHENG, R.; LIU, X. Cyber-physical systems: Milestones and research challenges. **Computer Communications**, v. 36, n. 1, p. 1–7, 2012. ISSN 0140-3664. Available at: <https://www.sciencedirect.com/science/article/pii/S0140366412003180>.

PUJO, P.; BROISSIN, N.; OUNNAR, F. PROSIS: An isoarchic structure for HMS control. **Engineering Applications of Artificial Intelligence**, v. 22, n. 7, p. 1034–1045, 2009. ISSN 09521976.

QUINTANILLA, F. G.; CARDIN, O.; L'ANTON, A.; CASTAGNA, P. Virtual Commissioning-Based Development and Implementation of a Service-Oriented Holonic Control for Retrofit Manufacturing Systems. In: BORANGIU, T.; TRENTESAUX, D.; THOMAS, A.; MCFARLANE, D. (Ed.). **Service Orientation in Holonic and Multi-Agent Manufacturing**. Cham: Springer, 2016, (Studies in Computational Intelligence, v. 640). p. 233–242. ISBN 978-3-319-30337-6.

ROS-INDUSTRIAL. Description. 2020. Available at: <https://rosindustrial.org/about/description>.

SCHUH, G.; ANDERL, R.; GAUSEMEIER, J.; HOMPEL, M. ten; WAHLSTER, W. **Industrie 4.0 Maturity Index**. 2017.

SCHWAB, K. **The Fourth Industrial Revolution**. New York: Currency Books, 2016.

SILVA, E. L.; MENEZES, E. M. **Metodologia da Pesquisa e Elaboração de Dissertação**. Florianópolis, Brasil: UFSC, 2005.

SIMÓN-MARMOLEJO, I.; RAMOS-VELASCO, L. Unified ontology for a holonic manufacturing system. **Revista Iberoamericana de Automática e Informática Industrial**, v. 15, p. 217–230, 2018.

SIMóN-MARMOLEJO, I.; LóPEZ-ORTEGA, O.; RAMOS-VELASCO, L.; ORTIZ-DOMíNGUEZ, M. Unified ontology for a holonic manufacturing system. **RIAI - Revista Iberoamericana de Automatica e Informatica Industrial**, v. 15, p. 217–230, 01 2018.

TAY, S.; CHUAN, L. T.; AZIATI, A.; AHMAD, A. N. A. An Overview of Industry 4.0: Definition, Components, and Government Initiatives. **Journal of Advanced Research in Dynamical and Control Systems**, v. 10, p. 14, 12 2018.

VALCKENAERS, P. Arti reference architecture – prosa revisited. v. 803, p. 1–19, 2018.

VDI. **Cyber-Physical Systems: Chancen und Nutzen aus Sicht der Automation**. Düsseldorf: VDI, 2013.

W3C. https://www.w3.org/standards/semanticweb/. 2015. Available at: <https://www.w3.org/standards/semanticweb/>.

WANG, L.; HAGHIGHI, A. Combined strength of holons, agents and function blocks in cyber-physical systems. **Journal of Manufacturing Systems**, v. 40, p. 25–34, 2016. ISSN 02786125.

WESER, M.; BOCK, J. D-2.1 Modulares Erweiterungskonzept der BaSys 4.0 Fähigkeitenontologie. 2020.

WOOLDRIDGE, M. J.; JENNINGS, N. R. **Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages, Amsterdam, the Netherlands, August 8-9, 1994. Proceedings**. 2. printing. ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. (Lecture Notes in Computer Science, v. 890). ISBN 3540588558. Available at: <http://www.springerlink.com/content/p7t30695h526>.

YANG, F.; GU, S. Industry 4.0, a revolution that requires technology and national strategies. **Complex & Intelligent Systems**, 2021. ISSN 2199-4536.

YASMEEN, G. Action research: An approach for the teachers in higher education. **The Turkish Online Journal of Educational Technology**, v. 7, n. 4, p. 46–53, 2008.

ZEZULKA, F.; MARCON, P.; VESELY, I.; SAJDL, O. Industry 4.0 – An Introduction in the phenomenon. **IFAC-PapersOnLine**, v. 49, n. 25, p. 8–12, 2016. ISSN 24058963.

A Appendix: Pump type A processing time

Table 19 – Pump Type A operation processing times.

Type A1	**Processing base time [s]**	**Type A2**	**Processing base time [s]**	**Type A3**	**Processing base time [s]**
A_1_1	720	A_2_1	900	A_3_1	1080
A_1_2	600	A_2_2	750	A_3_2	900
A_1_3	1320	A_2_3	1650	A_3_3	1980
A_1_4	3600	A_2_4	4500	A_3_4	5400
A_1_5	1800	A_2_5	2250	A_3_5	2700
A_1_6	1080	A_2_6	1350	A_3_6	1620
A_1_7	2160	A_2_7	2700	A_3_7	3240
A_1_8	1320	A_2_8	1650	A_3_8	1980
A_1_9	1140	A_2_9	1425	A_3_9	1710
A_1_10	1500	A_2_10	1875	A_3_10	2250
A_1_11	1320	A_2_11	1650	A_3_11	1980
A_1_12	2100	A_2_12	2625	A_3_12	3150
A_1_13	3600	A_2_13	4500	A_3_13	5400
A_1_14	1320	A_2_14	1650	A_3_14	1980
A_1_15	3600	A_2_15	4500	A_3_15	5400
A_1_16	0	A_2_16	0	A_3_16	0
Total [h]	**7.6**	**Total [h]**	**9.4**	**Total [h]**	**11.3**

B Appendix: Pump type B processing time

Table 20 – Pump Type B operation processing times.

Type B1	**Processing base time [s]**	**Type B2**	**Processing base time [s]**	**Type B3**	**Processing base time [s]**
B_1_1	270	B_2_1	337.5	B_3_1	405
B_1_2	150	B_2_2	187.5	B_3_2	225
B_1_3	150	B_2_3	187.5	B_3_3	225
B_1_4	90	B_2_4	112.5	B_3_4	135
B_1_5	90	B_2_5	112.5	B_3_5	135
B_1_6	210	B_2_6	262.5	B_3_6	315
B_1_7	150	B_2_7	187.5	B_3_7	225
B_1_8	30	B_2_8	37.5	B_3_8	45
B_1_9	120	B_2_9	150	B_3_9	180
B_1_10	120	B_2_10	150	B_3_10	180
B_1_11	120	B_2_11	150	B_3_11	180
B_1_12	360	B_2_12	450	B_3_12	540
B_1_13	60	B_2_13	75	B_3_13	90
B_1_14	270	B_2_14	337.5	B_3_14	405
B_1_15	60	B_2_15	75	B_3_15	90
B_1_16	180	B_2_16	225	B_3_16	270
B_1_17	180	B_2_17	225	B_3_17	270
B_1_18	150	B_2_18	187.5	B_3_18	225
B_1_19	300	B_2_19	375	B_3_19	450
B_1_20	0	B_2_20	0	B_3_20	0
Total [h]	**0.9**	**Total [h]**	**1.1**	**Total [h]**	**1.3**

C Appendix: Pump type C processing time

Table 21 – Pump Type C operation processing times.

Type C1	**Processing base time [s]**	**Type C2**	**Processing base time [s]**	**Type C3**	**Processing base time [s]**
C_1_1	600	C_2_1	750	C_3_1	900
C_1_2	1200	C_2_2	1500	C_3_2	1800
C_1_3	1200	C_2_3	1500	C_3_3	1800
C_1_4	1200	C_2_4	1500	C_3_4	1800
C_1_5	1200	C_2_5	1500	C_3_5	1800
C_1_6	1200	C_2_6	1500	C_3_6	1800
C_1_7	1200	C_2_7	1500	C_3_7	1800
C_1_8	600	C_2_8	750	C_3_8	900
C_1_9	1200	C_2_9	1500	C_3_9	1800
C_1_10	1800	C_2_10	2250	C_3_10	2700
C_1_11	1200	C_2_11	1500	C_3_11	1800
C_1_12	1200	C_2_12	1500	C_3_12	1800
C_1_13	600	C_2_13	750	C_3_13	900
C_1_14	1800	C_2_14	2250	C_3_14	2700
C_1_15	0	C_2_15	0	C_3_15	0
Total [h]	**4.5**	**Total [h]**	**5.6**	**Total [h]**	**6.8**