Leonardo is among the top ten global players in Aerospace, Defence and Security and Italy’s main industrial company. It is organised into seven business divisions.

Listed on the Milan Stock Exchange (LDO), in 2017 Leonardo recorded consolidated restated revenues of 11.7 billion Euros and has a significant industrial presence in Italy, the UK, the US and Poland.
Airborne Radar & Advanced Targeting
Organisational Structure

Model-Based Design
Functional Organisational Structure

Systems
Software
Firmware
Organisational structure reflected in processes

Systems

MATLAB Model

Algorithm Document

Implementation

Software

Firmware

Or
Legacy issues

Organisational structure reflected in processes

Systems

MATLAB Model

Algorithm Document

Implementation

Software

Firmware

Or
Legacy issues

Organisational structure reflected in processes

Systems

MATLAB Model

→

Algorithm Document

→

Implementation

Software

Or

Firmware
Multiple points of failure

Error in MATLAB model
Multiple points of failure

Error in MATLAB model

Error translating to document
Multiple points of failure

Error in MATLAB model

Error translating to document

Error interpreting document
Multiple points of failure

- Error in MATLAB model
- Error translating to document
- Error interpreting document
- Error implementing document
Advantages:
+ Optimised Software/Firmware implementations
+ Established process supported by experienced engineers

Disadvantages:
- Extensive multi stage reviews
- Slow multi-function iteration cycles
- Independent multi stage testing
- Extensive documentation
- Limited collaboration of solution
- Targeting hardware late in lifecycle
Advantages:
+ Optimised Software/Firmware implementations
+ Established process supported by experienced engineers

Disadvantages:
- Extensive multi stage reviews
- Slow multi-function iteration cycles
- Independent multi stage testing
- Extensive documentation
- Limited collaboration of solution
- Targeting hardware late in lifecycle
Reduced points of failure

Model-Based Design in Simulink

Auto-generated Code
Introducing MDEMR

Model Based Design is not new at Leonardo Edinburgh and has been used for over 10 years although technology and design toolset advances present new opportunities:

- Dynamic and intuitive engineering process updates
- Expansion of advanced infrastructure
- Knowledge sharing leading to widespread adoption
- Increase cross-functional collaboration at model level (Systems/Software/Firmware)
Introducing MDEMR

Model Based Design is not new at Leonardo Edinburgh and has been used for over 10 years although technology and design toolset advances present new opportunities:

- Dynamic and intuitive engineering process updates
- Expansion of advanced infrastructure
- Knowledge sharing leading to widespread adoption
- Increase cross-functional collaboration at model level (Systems/Software/Firmware)

**Challenge:** How do we **scale** Model-Based Design to realise these opportunities?
Model Driven Engineering, Modularity & Re-use
Cross Functional

New Technology

Development Tools

Academic Placements

MDE Process

Reference Designs

MDEMR

MDEMR Process
Cross-functional team containing **specialist engineers** dedicated to addressing Model-Based Design capability at Leonardo.

- Working with Systems, Software and Firmware to ensure MBD is not counter to
  - Existing processes
  - Development environments

- Leverages the **full lifecycle** capability of the MathWorks toolset

- Drive continuous improvement and best practice
Cross Functional

New Technology

Academic Placements

Reference Designs

Development Environment

MDEMR Process

MDEMR Reference Designs

New Technology

Academic Placements

Reference Designs

Development Environment
Infrastructure for Model-Based Design

Fundamental to developing complex multi-functional models is to have a development environment capable of supporting high integrity designs in collaboration.

Common MATLAB & Simulink Workflows:

- Issue Management
- Source Control
- Test Automation
Common Environment

Mandate the use of **Simulink Projects** for both MATLAB and Simulink designs

- Standardised environment setup
  - No more ‘add all to path... then load this file... but not that one’
- Use project **Templates** to **distribute** standardised projects
- Reflect model architecture using **Referenced Projects**
- Source Control integration
Source Control

Migrated to GIT from legacy source control solution

- Enables collaborative branching workflows which are not file locking based
- GitFlow for its scalability and traceability
Source Control

Migrated to GIT from legacy source control solution

- Enables collaborative branching workflows which are **not** file locking based

- GitFlow for its scalability and traceability

How does a branching workflow work for Simulink?
GitFlow for Model-Based Design

Only possible due to the excellent advancements in merge and diff tool capability of Simulink models.

Fundamental to success is communication and model componentisation.

- Model updates must be planned and scoped - branch cannot be open indefinitely
- Model must be well structured: Referenced Models and Libraries

How does a branching workflow work for Simulink?
Automated Testing

Investing in automated **build** and **test** pipelines for Simulink

Automates generation of auto-code (C/C++)

Verification and Validation of model

Verification and Validation of auto-generated code.
Dynamic Model-Driven Engineering (MDE) Process

Process that defines how to develop Model-Based Designs in Simulink

- Rapid prototyping
- Main development and modelling
- Partitioning to Software/Firmware (Fixed Point)
- Targeting representative hardware through PIL, FIL and SysIL testing

Leverage the cross-discipline expertise.
Model-Driven Engineering Process

Uses **Live Editor** to give interactive examples on each step that leverage internal referenced designs e.g.

- How to use Test Manager for SIL/PIL/FIL equivalence testing of requirements?
- How to setup environment? e.g.
  - Git repo
  - Simulink Projects
  - Jenkins
- How to deploy?
Referenced Designs

A key factor in scaling and promoting best practice to the Leonardo engineering community is through referenced designs which are published internally.

Referenced designs are relevant to Leonardo products to better engage with user base e.g. Radar and tracking algorithms.

Referenced designs are used to investigate new technologies and promote re-use.

Referenced designs are configured to showcase stages of MDE Process and lifecycle.
MDEMR

Cross Functional

New Technology

Development Environment

Academic Placements

MDE Process

Reference Designs
Academic Placements

Recruitment

Technology
Academic Placements

- Offer exciting 3, 6 and 12 month placements
- Individual deliverable projects that ties in with MDE strategy
- Wider business exposure
- Pave the way for future work and employment
Academic Placements

- Offer **exciting** 3, 6 and 12 month placements
- Individual deliverable projects that ties in with MDE strategy
- Wider business exposure
- Pave the way for future work and employment

Recruitment

- Investigate new emerging technologies
- Trial and feedback on MATLAB and Simulink pre-releases
- Develop reference designs showcasing **best practice**

Technology
Examples of Summer placement work with MATLAB

Deep Learning for Object Detection

Targeting embedded GPUs with GPU Coder
Deep Learning for Object Detection

- Trained on 5000 ‘simple’ target images
- Uses FasterRCNN MATLAB implementation
- Accurately predicts targets in cluttered environment
Targeting embedded GPUs with GPU Coder

- Auto generates CUDA from m-code using GPU Coder
- Runs on target Jetson TX2 embedded hardware
- Closer to a real-world implementation

SAR image formation

Jetson TX2
Academic Placements

- Offer **exciting** 3, 6 and 12 month placements
- Individual deliverable projects that ties in with MDE strategy
- Wider business exposure
- Pave the way for future work and employment

Recruitment

- Investigate new emerging technologies
- Trial and feedback on MATLAB and Simulink pre-releases
- Develop reference designs showcasing **best practice**

Technology

Reference Designs
What’s a GPU?

Graphical Processing Unit (GPU)
- Originally for graphical processing for video and games

Highly Parallel Architecture
- Many thousands of computing cores
- Capable of spawning many threads
- Allows for massive parallelism in code

Plug and Play
- Cards can be inserted into the PCIe slot on most motherboards

Simulation Acceleration
- Powerful for tackling compute intensive mathematical modelling
- Can provide order of magnitude speedups over CPU implementations
What’s a GPU?

Graphical Processing Unit (GPU)
- Originally for graphical processing for video and games

Highly Parallel Architecture
- Many thousands of computing cores
- Capable of spawning many threads
- Allows for massive parallelism in code

Plug and Play
- Cards can be inserted into the PCIe slot on most motherboards

Simulation Acceleration
- Powerful for tackling compute intensive mathematical modelling
- Can provide order of magnitude speedups over CPU implementations
What's a GPU?

**Graphical Processing Unit (GPU)**
- Originally for graphical processing for video and games

**Highly Parallel Architecture**
- Many thousands of computing cores
- Capable of spawning many threads
- Allows for massive parallelism in code

**Plug and Play**
- Cards can be inserted into the PCIe slot on most motherboards

**Simulation Acceleration**
- Powerful for tackling compute intensive mathematical modelling
- Can provide order of magnitude speedups over CPU implementations
What’s a GPU?

Graphical Processing Unit (GPU)
- Originally for graphical processing for video and games

Highly Parallel Architecture
- Many thousands of computing cores
- Capable of spawning many threads
- Allows for massive parallelism in code

Plug and Play
- Cards can be inserted into the PCIe slot on most motherboards

Simulation Acceleration
- Powerful for tackling compute intensive mathematical modelling
- Can provide order of magnitude speedups over CPU implementations
Simulation Acceleration

Parallel Computing Toolbox

% Filthy elementwise multiply
X = A.*B;

% Nasty matrix inversion
Y = inv(X);

% Disgusting fft
fft_y = fft(Y);
Examples

Aircraft radome antenna modelling

Radar beam forming

Synthetic Aperture Radar (SAR)

20x speedup
Examples

Aircraft radome antenna modelling

Radar beam forming

Synthetic Aperture Radar (SAR)

20x speedup

50x speedup
Examples

Aircraft radome antenna modelling

Radar beam forming

Synthetic Aperture Radar (SAR)

20x speedup

50x speedup

100x speedup
Examples

Aircraft radome antenna modelling

Radar beam forming

Synthetic Aperture Radar (SAR)

How do we use a GPU in a real-world environment?

20x speedup

50x speedup

100x speedup
GPU Coder -> Deploy to Hardware -> Straight into product
Future Use

GPU Coder → Deploy to Hardware → Straight into product
GPU Coder → Deploy to Hardware → Straight into product