Centre for Autonomous and Cyber-Physical Systems
School of Aerospace, Transport and Manufacturing

Marcin Biczyski
m.biczyski@cranfield.ac.uk

2017-2018 AVDC MSc Individual Research Project:

Domain Specific Language for Drone Swarm Evasive Manoeuvring

Viva voce

3rd September 2018

Supervisors: Prof Żbikowski and Dr Savvaris

Marcin Biczyski
m.biczyski@cranfield.ac.uk

Project definition and contributions

Project definition:

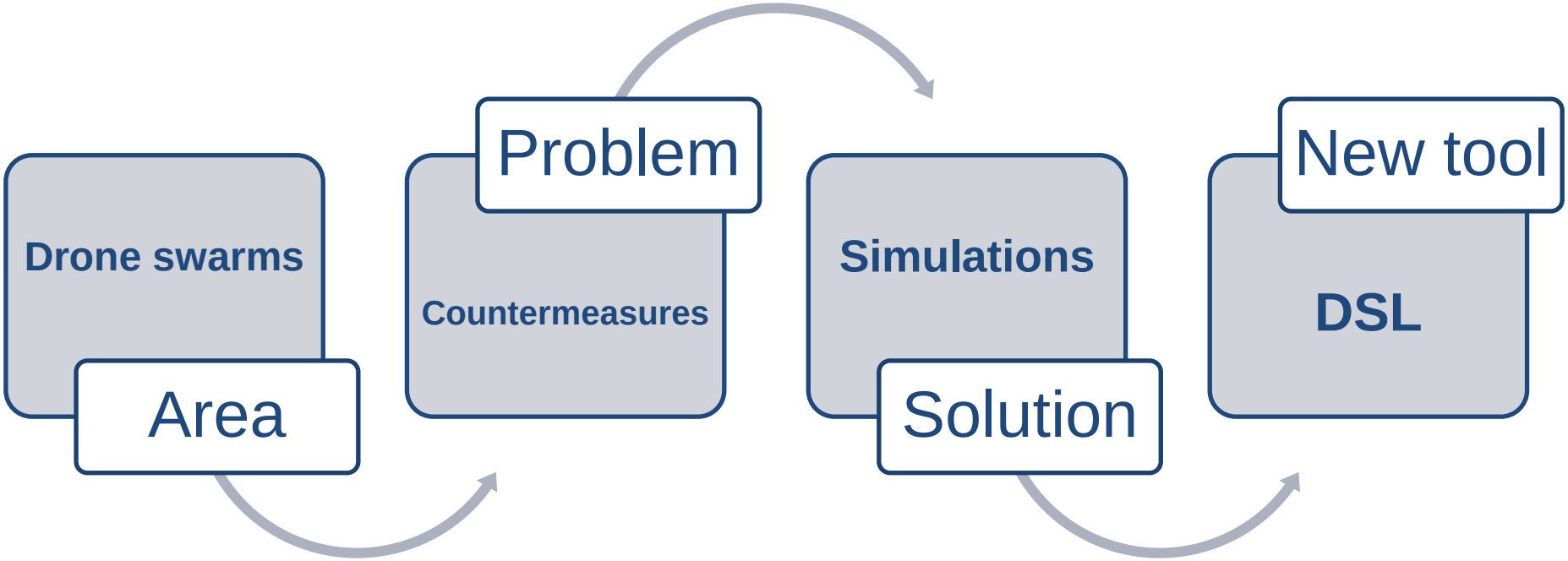
The project focuses on showcasing Domain Specific Languages (DSLs) as a tool for simulations in complex domains capable of emergent behaviours. A complete process is shown, from methodology through DSL design to validation by an example application, that results in a tool for optimisation of drone swarm behaviours in the presence of countermeasures.

Contributions:

The work has shown the possibility of transforming a Framework Specific methodology to suit different domains. Additionally, a novel DSL was developed for use with drone swarms and threats they might encounter.

Marcin Biczyski
m.biczyski@cranfield.ac.uk

Motivation and background



Marcin Biczyski
m.biczyski@cranfield.ac.uk

Problem definition

Aims:

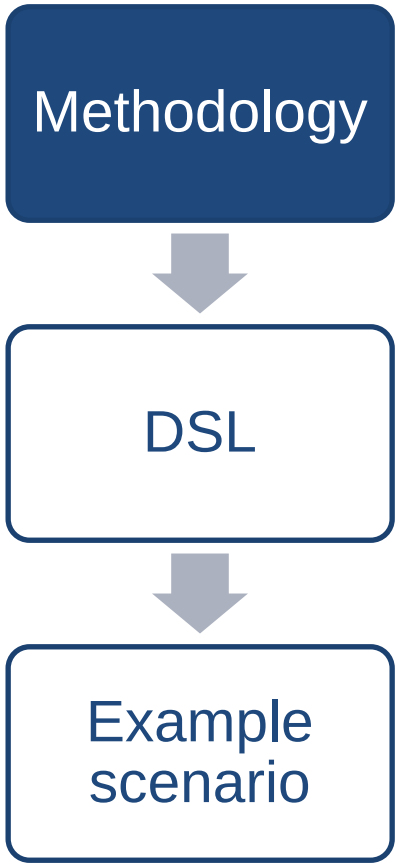
- Show that a DSL can be an **useful tool** for modelling and simulation engineering,
- Introduce **modularity** to drone swarm simulations,
- Present a methodology for creation of DSLs in a domain capable of **emergent behaviours**.

Objectives:

- Accommodate an existing DSL creation methodology for the domain,
- Create a DSL for drone swarm simulations,
- Validate the DSL for drone swarm path optimisation.

Marcin Biczyski
m.biczyski@cranfield.ac.uk

Solution approach



Iteration 1

- Adopted Framework Specific methodology
- Added ‘Methodology evaluation’ phase

Iteration 2

- Removed inapplicable steps
- Added Framework for Qualitative Assessment of DSLs

Iteration 3

- Added 2 new steps
- Finalised methodology

Marcin Biczyski
m.biczyski@cranfield.ac.uk

Solution approach

Methodology

DSL

Example scenario

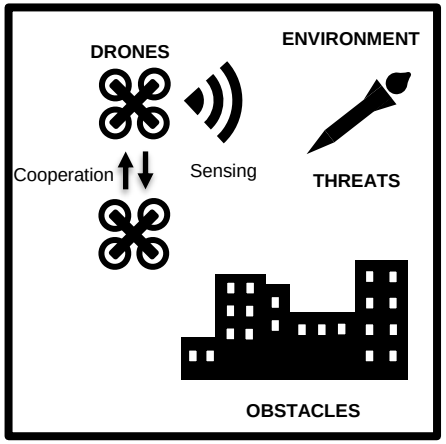
Domain

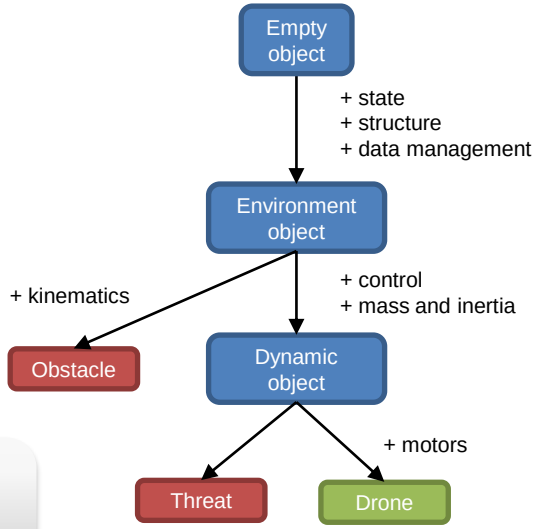
Domain Specific Model

Simulation

Constraints

Optimisation





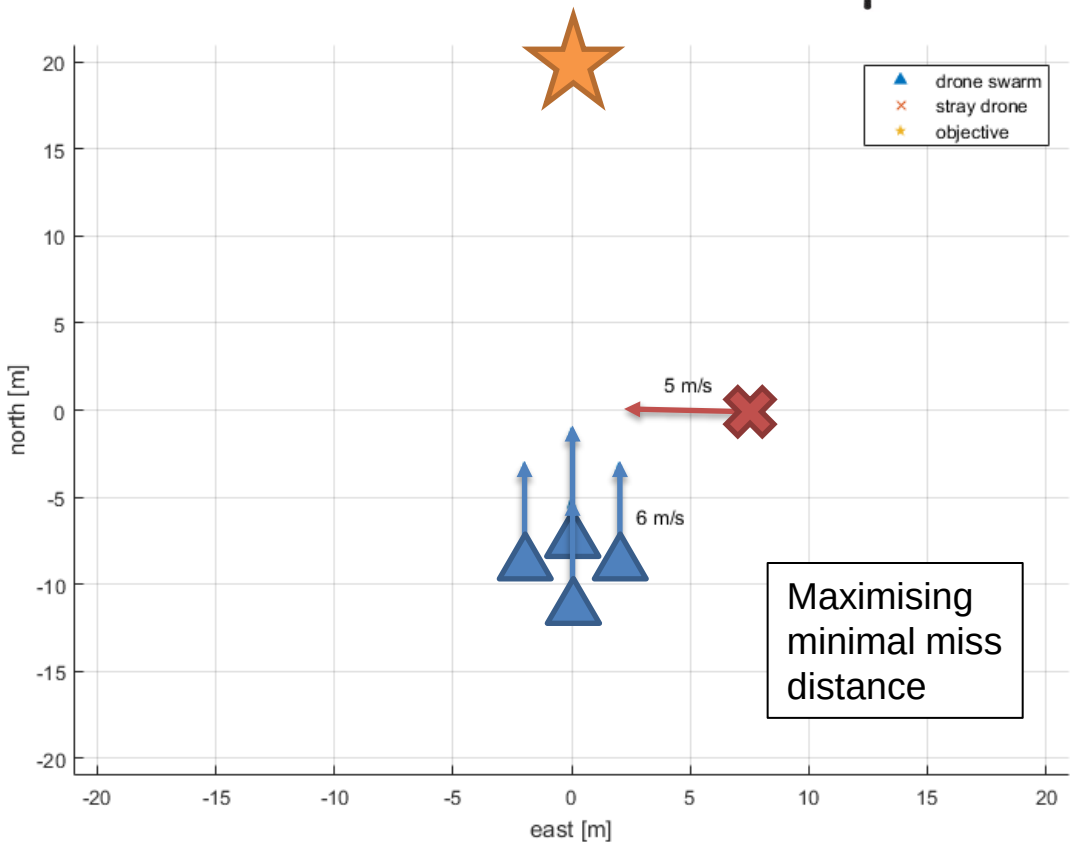
Marcin Biczyski
m.biczyski@cranfield.ac.uk

Solution approach

Methodology

DSL

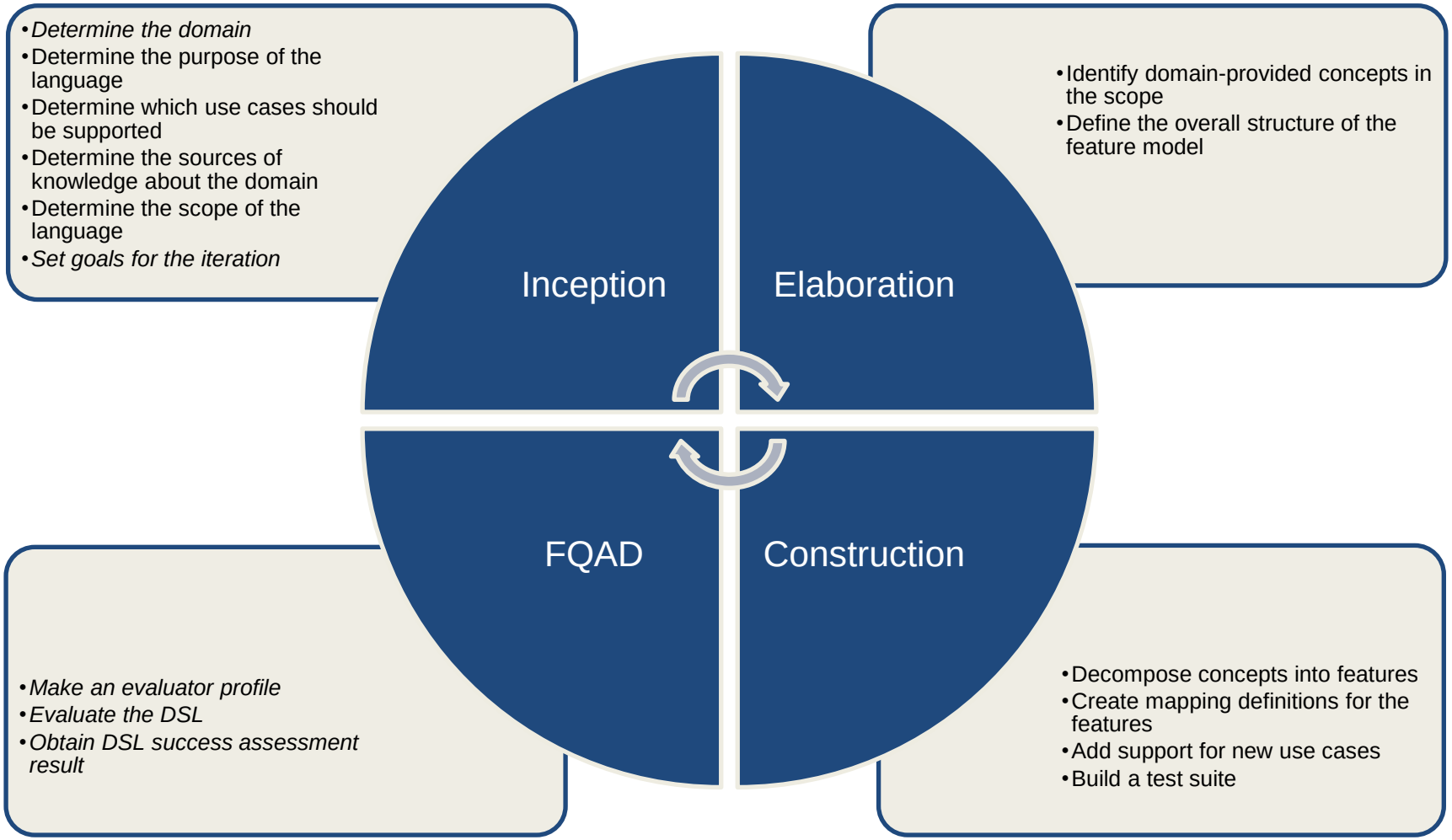
Example scenario



Method: Simulated annealing

Marcin Biczyski
m.biczyski@cranfield.ac.uk

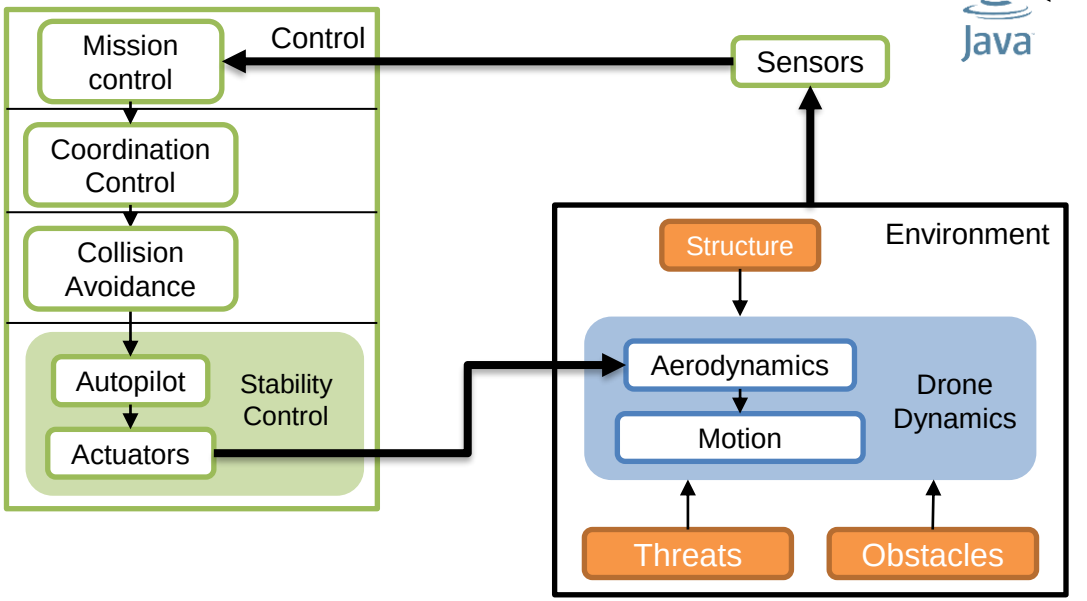
Results - methodology



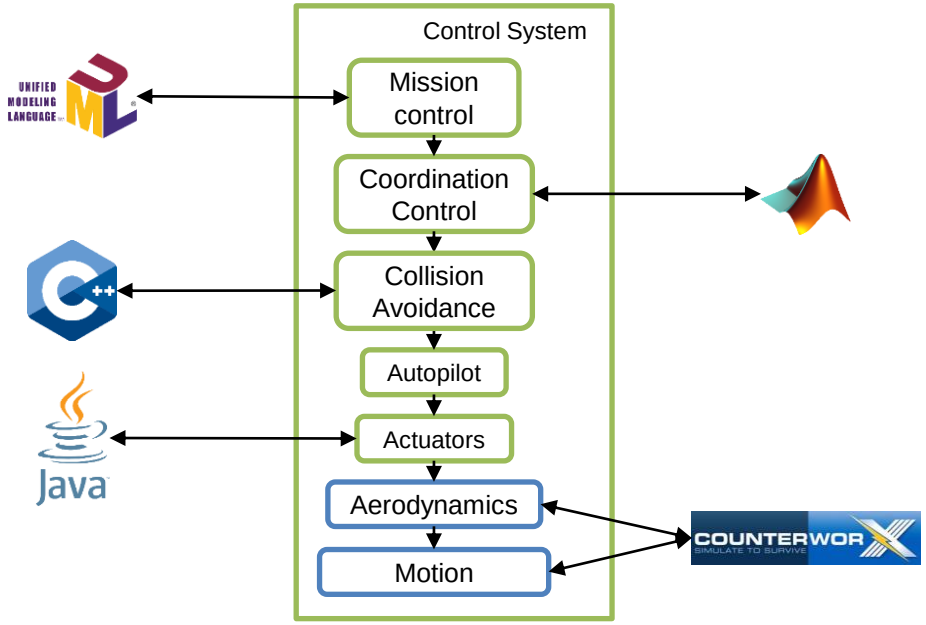
Marcin Biczyski
m.biczyski@cranfield.ac.uk

Results - DSL

Layered Control System

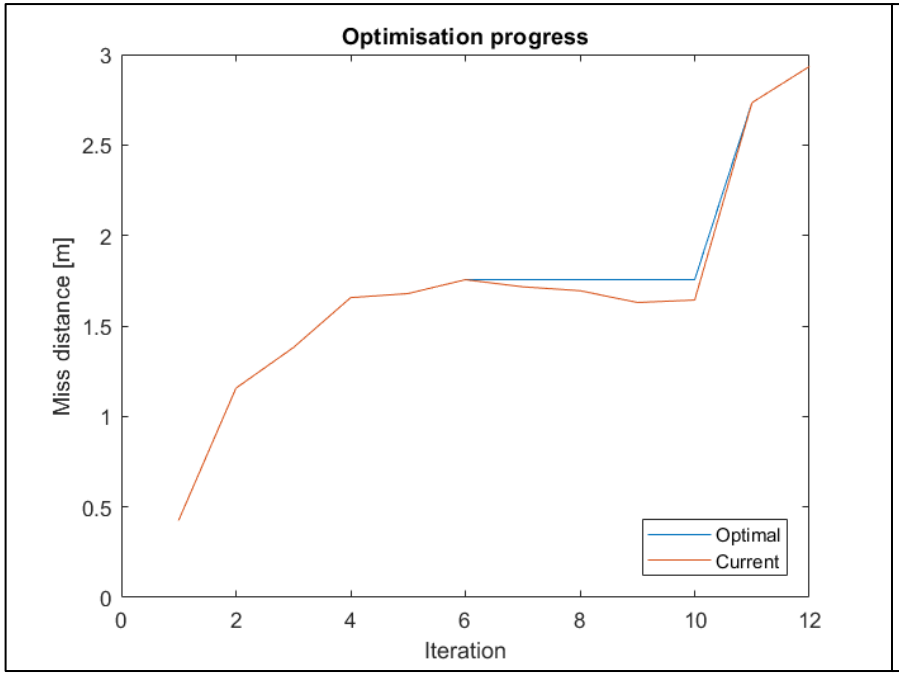


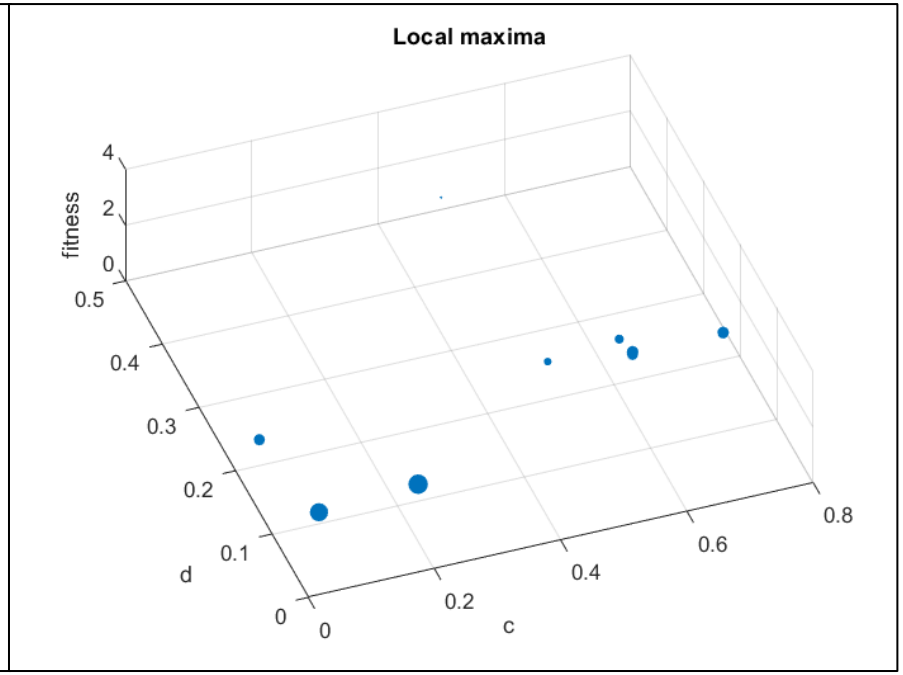
Modular Code Building



Marcin Biczyski
m.biczyski@cranfield.ac.uk

Results – example scenario





Method used: **simulated annealing**

Parameters	Temperature	Attraction c	Momentum d	Repulsion r	Objective g
Initial	10	0.5	0.5	1	0.01
Final	2.79	0.179	0.009	1.367	0.01

Marcin Biczyski
m.biczyski@cranfield.ac.uk

Discussion of results – language comparison

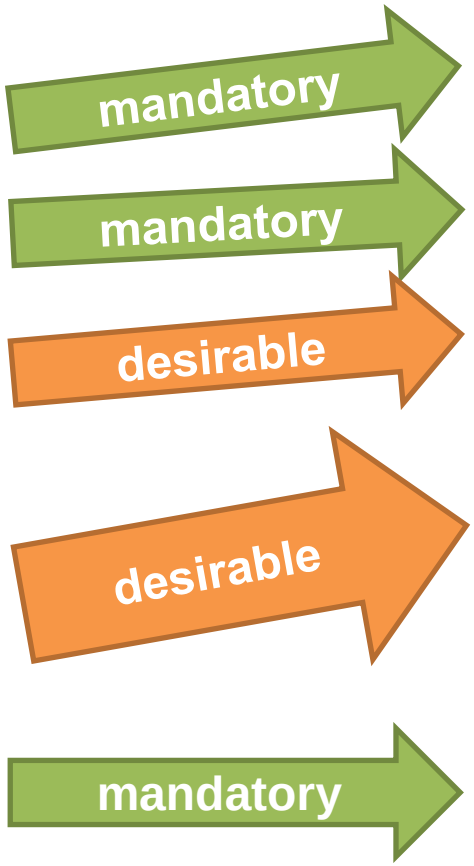
Language	Drone Swarms DSL	Applet FSML
Developer, year	Marcin Biczyski, 2018	Michał Antkiewicz, 2006
Domain	3D aerospace with multiple aircraft and obstacles	Java 5.0 Applet
Value proposition	(1) Provide drone swarm path generation utility; (2) Enable modelling of threats and obstacles in the domain; (3) Provide utility for performance measurement and optimisation of generated paths; (4) Ease the process of simulation and optimisation	(1) Drive the extension of the generic FSML infrastructure to support defining FSMLs declaratively; (2) Compare FSML approach to design fragments approach using applets as benchmark; (3) Demonstrate feasibility of round-trip engineering; (4) Provide an introductory FSML example
Use cases	Manoeuvre Modelling; Decentralised Decision Making; Constraints Specification; Model Parametrisation; Path Optimisation; Modular Synthesis	Framework API understanding; Completion code analysis and understanding: understanding, API constraint checking; Completion code creation; Completion code evolution
Horizontal scope	drones, threats, manoeuvres, optimisation, constraints	applets, status, mouse listeners, threads, parameters
Artefact type	Racket (Lisp)	Java
Sources of knowledge	articles, expert knowledge (pilots and operators), existing models	API, tutorials, articles, expert knowledge, example applications

Centre for Autonomous and Cyber-Physical Systems
School of Aerospace, Transport and Manufacturing

Marcin Biczyski
m.biczyski@cranfield.ac.uk

Discussion of results – language validation

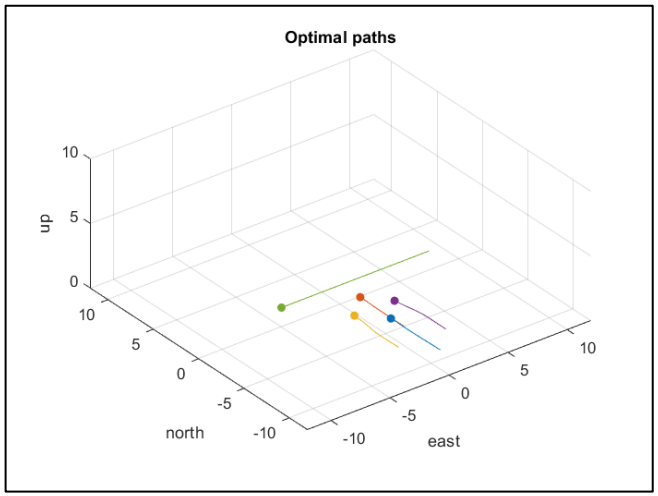
DSL Success Assessment Questionnaire		
Please give marks to every sentence below for the assessment of the DSL.		
State how much you agree to each sentence by ticking the appropriate choice.		
	DSL Success Quality Measures	**Support Level**
	Functional Suitability	
1.	All concepts and scenarios of the domain can be expressed in the DSL (completeness)	Strong support
2.	DSL is appropriate for the specific applications of the domain (e.g. to express an algorithm) (appropriateness)	Strong support
	Usability	
3.	The required amount of effort for understanding the language is small (comprehensibility)	Strong support
4.	The concepts and symbols of the language are easy to learn and remember (learnability)	Strong support
5.	Language has capability to help users achieve their tasks in a minimum number of steps	Strong support
6.	Users can recognize whether the DSL is appropriate for their needs (likeability,user perception)	Full support
7.	DSL has attributes that make it easy to operate and control the language (operability)	Strong support
8.	DSL has symbols that are good-looking (attractiveness)	Strong support
9.	The language provides mechanisms for compactness of the representation of the program. (compactness)	Strong support
	Reliability	
10.	DSL protects users against making errors. The DSL avoids the user to make mistakes. (model checking)	Some support
11.	DSL includes right elements and correct relations between them (DSL prevents the unexpected interactions between its elements) (correctness)	Some support
	Maintainability	
12.	The amount of effort required for modifying the DSL to provide different or additional functionality is small (modifiability)	Some support
13.	DSL is composed of discrete components such that a change to one component has minimal impact on other components (Low coupling)	Some support
	Productivity	
14.	The development time of a program to meet the needs is improved	Strong support
15.	The amount of human resource used to develop the program is improved	Strong support
	Extensibility	
16.	DSL has general mechanisms for users to add new features (adding new features without changing the original language)	Strong support
	Compatibility	
17.	DSL is compatible with the domain. DSL has capability to operate with other elements of the domain with no modification required to perform a specific application in the domain.	Some support
18.	Using DSL to develop models fits in the development process, since it is used as part of a development process with phases and roles.	Some support
	Expressiveness	
19.	A problem solving strategy can be mapped into a program easily	Strong support
20.	The DSL that provides one and only one good way to express every concept of interest (unique)	Strong support
21.	Each DSL construct is used to represent exactly one distinct concept in the domain (orthogonal)	Strong support
22.	The language constructs correspond to important domain concepts. DSL does not include domain concepts that are not important.	Full support
23.	DSL does not contain conflicting elements.	Strong support
24.	DSL is at the right abstraction level such that it is not more complex or detailed than necessary	Strong support
	Reusability	
25.	The symbols and other elements of the DSL can be used in more than one DSL, or in building other language elements. Using the definition of a language as a beginning to develop a new one (Reusability)	Strong support
	Integrability	
26.	DSL can be integrated with other language used in development process. (language integrability with other languages)	Strong support

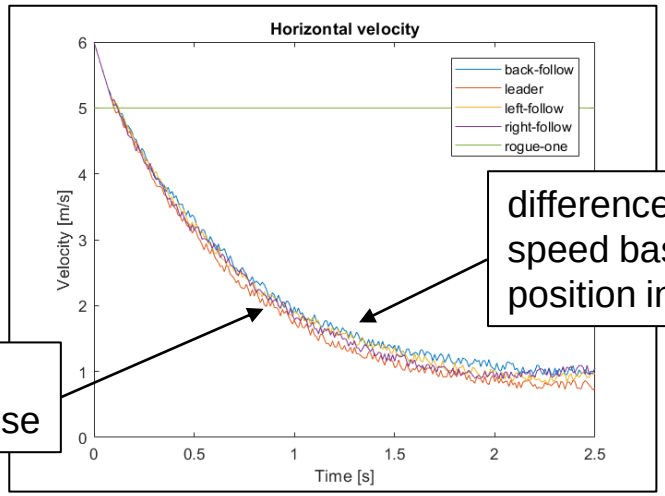


DSL Success Assessment Results		
Functional Suitability		
	Completeness	SATISFACTORY
	Appropriateness	SATISFACTORY
Usability		
	Comprehensibility	SATISFACTORY
	Learnability	SATISFACTORY
	Language helps users achieve their tasks in a minimum number of steps	SATISFACTORY
	Likeability,user perception	EFFECTIVE
	Operability	SATISFACTORY
	Attractiveness	SATISFACTORY
	Compactness	SATISFACTORY
Reliability		
	Model checking ability	SATISFACTORY
	Correctness	SATISFACTORY
Maintainability		
	Modifiability	EFFECTIVE
	Low coupling	EFFECTIVE
Productivity		
	The development time improvement	EFFECTIVE
	The amount of human resource used improvement	EFFECTIVE
Extensibility		
	Mechanisms for users to add new features	EFFECTIVE
Compatibility		
	DSL is compatible to the domain	SATISFACTORY
	Using DSL to develop models fits in the development process	SATISFACTORY
Expressiveness		
	A problem solving strategy can be mapped into a program easily	SATISFACTORY
	Uniqueness	SATISFACTORY
	Orthogonality	SATISFACTORY
	The language constructs correspond to important domain concepts.	EFFECTIVE
	DSL does not contain conflicting elements.	SATISFACTORY
	DSL is at the right abstraction level	SATISFACTORY
Reusability		
	Reusability	EFFECTIVE
Integrability		
	Integrability	EFFECTIVE
	OVERALL SUCCESS OF THE DSL	EFFECTIVE

Centre for Autonomous and Cyber-Physical Systems
School of Aerospace, Transport and Manufacturing
Marcin Biczyski
m.biczyski@cranfield.ac.uk

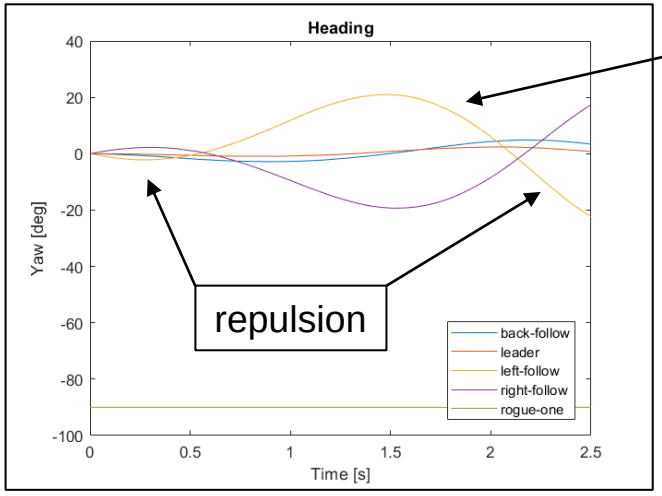
Discussion of results – behaviour analysis





difference in speed based on position in swarm

visible angular noise



attraction

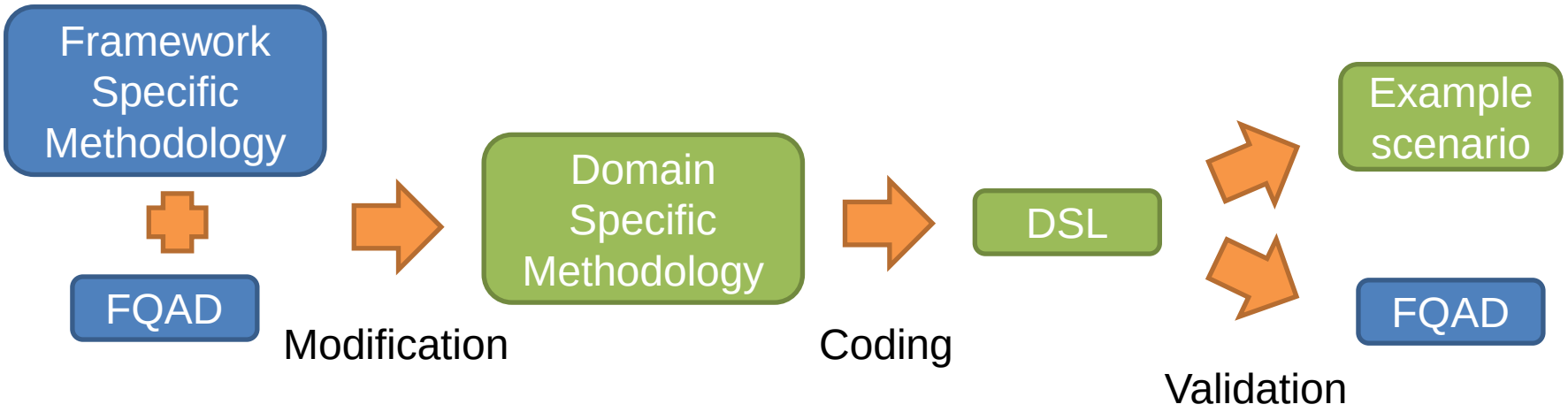
repulsion

Best behaviour found is **sub-optimal!**

- Error associated with cooling procedure
- Not enough iterations
- Low initial temperature

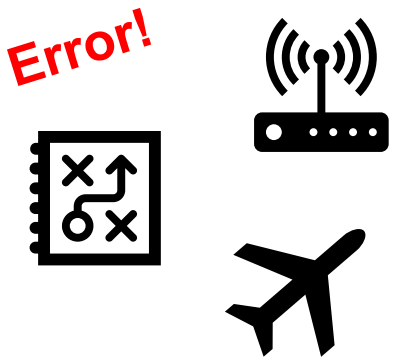
Marcin Biczyski
m.biczyski@cranfield.ac.uk

Conclusions and future work



Future work:

- Error-proofing - contracts,
- Drone communication system,
- Graphical data representation,
- Fixed-wing UAV support.



Marcin Biczyski
m.biczyski@cranfield.ac.uk

Thank you!