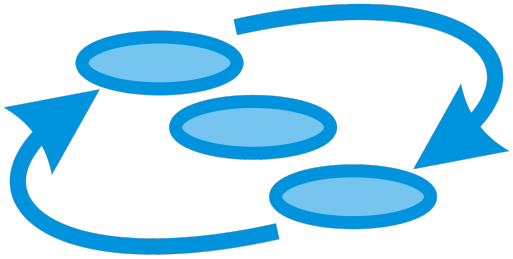
Grant Agreement: 644047

INtegrated TOol chain for model-based design of CPSs



DSE in the INTO-CPS Platform

Deliverable Number: D5.1a

Version: 1.0

Date: 2015

Public Document

http://into-cps.au.dk

Contributors:

Carl Gamble, UNEW
Richard Payne, UNEW
Claes Dühring Jæger, AI
Francois Hantry, CLE
Christian König, TWT
Alie El-din Madie, UTRC

Editors:

Carl Gamble, UNEW

Reviewers:

Francois Hantry, CLE
Andrey Sadovykh, ST
Claes Dühring Jaeger, AI

Consortium:

Aarhus University	AU	Newcastle University	UNEW
University of York	UY	Linköping University	LIU
Verified Systems International GmbH	VSI	Controllab Products	CLP
ClearSy	CLE	TWT GmbH	TWT
Agro Intelligence	AI	United Technologies	UTRC
Softeam	ST		

Document History

Ver	Date	Author	Description
0.1	22-05-2015	Carl Gamble	Initial document version
0.2	26-10-2015	Carl Gamble	Content from iD3.1 added
0.3	13-11-2015	Alie El-Din Mady	UTRC DSE case study added
0.4	14-11-2015	Claes Jæger	AI case study added
0.5	15-11-2015	Carl Gamble	Python scripts added
0.6	16-11-2015	Carl Gamble	Progress and plans added
0.7	14-12-2015	Carl Gamble	AI and ST review comments addressed
1.0	15-12-2015	Carl Gamble	Internal review comments addressed

Abstract

This deliverable presents the outputs of Tasks 3.2 and 5.1 together, including both the contributions of the methodology and tool creation activities. The methodology reports upon the aspirations of the industry partners with respect to DSE in their case studies before outlining the DSE approaches that are planned to be implemented during the project. The tool creation activity reports on the design of the DSE module, focussing on its separation into components and their interaction before presenting an outline plan for their development.

Contents

1 Introduction

This is the first deliverable concerning the DSE tasks in INTO-CPS. It is designated as D5.1a and so represents the output of the DSE module tool task, T5.1, but for conciseness it also includes the methodological output of T3.2.

The document begins with a statement of the state of the art in DSE in CPSs and then moves on to a description of key aspects of the INTO-CPS case studies as described by the case study owners with some key points drawn out in Section 3. Section 4 outlines the proposed methods that the DSE module should support and Section 5 presents an outline of how the DSE module works, its structure, the progress to date and a plan for the future development.

2 Related Work

In the *DESTECS* project[1] DSE was supported by applying Automated Co-model Analysis (ACA), such as parameter sweep. The project also provided support for testing different model implementations. The project provided methodological guidelines for DSE in [BFG+12] and [FLPV13] and tool support for the Crescendo in the form of ACA [NBAR+12]. INTO-CPS will use the methods work from DESTECS as a baseline, extended with wider range of analysis techniques and including closed loop support.

The *Certainty* project[2] uses DSE in the DOL-Critical method; using the results of interference analysis reliability analysis to evaluate potential mapping and scheduling solutions of tasks to cores on multi-core platforms. The project includes several tools: "the EXPO tool is the central module of the framework. As an underlying multi-objective search algorithm, the Strength Pareto Evolutionary Algorithm (SPEA2) is used that communicates with EXPO via the PISA interface" [CER13a]. The project also proposes the "Mixed Criticality Mapping and Scheduling Optimisation (MCMSO) method" which implements a heuristic method based on simulated annealing [CER13b]. Both the implementation and methods developed by Certainty will influence DSE in INTO-CPS, in particular the use of simulated

[1] `http://www.destecs.org/`

[2] `http://www.certainty-project.eu/`

annealing and Pareto Front techniques. It is not clear to what extent this work is 'closed-loop', one focus of DSE in INTO-CPS.

As part an integrated tool chain for high-level synthesis of high-performance FPGA systems, the *ENOSYS*[3] project uses two tools for DSE: *FalconML*[4] and *Jink*[5]. The Jink Design Space Explorer coordinates a design flow and its exploration engine searches over various parameters used in customising the soft core multi processor and in partitioning the UML design to the underlying architecture. Jink finally parses over the various logs and reports files produced by the various tools during synthesis, compilation, simulation to extract various design characteristics and metrics. This work is limited to FPGA design and it is not clear to what extent this work is 'closed-loop', or which ranking or analysis methods are used. Further investigation is required to determine the extent of influence these outputs may have on INTO-CPS.

The ongoing *AXIOM*[6] project will provide DSE, with a deliverable on DSE due in M24 of the project (January 2017). DSE technologies will be used in the development of the software parts and the selection of the most appropriate hardware architecture and interconnect.

MADNESS[7] use traditional methods for DSE, alongside their "co-exploration" which uses different search algorithms for different dimensions and they report that "multidimensional co-exploration can find better design points and evaluates a higher diversity of design alternatives as compared to the more traditional approach of using a single search algorithm for all dimensions." We have been unable to obtain public deliverables.

In the *iCyPhy* project[8], effort is placed to reduce the design space for DSE through optimal architecture selection [FNSV15]. A routine is defined to optimise the continuous parameters of a CPS to decrease the number of simulations.

The *DARPA AVM META* project[9] defines the CyPhyML for the modelling of CPSs. The project uses the Design Space Exploration Tool (DESSERT) to prune the design space to a "manageable size". INTO-CPS should consider the DESSERT technology and its methods for design space reduction.

[3] https://sites.google.com/a/enosys-project.eu/www/home
[4] https://sites.google.com/a/enosysproject.eu/www/enosys-tools/falconml
[5] https://sites.google.com/a/enosys-project.eu/www/enosys-tools/jink
[6] http://www.axiom-project.eu
[7] http://www.madnessproject.org/
[8] http://www.icyphy.org/index.html
[9] http://cps-vo.org/group/avm/meta

The Merlin project[10] produced a suit of tools collectively called the Strategic Decision Making Tool (SDMT) to facilitate exploring the design space within the rail domain. This suite consists of a core tool that orchestrates the DSE, an optimisation tool responsible for driving the DSE through the use of a genetic algorithm and pareto optimality analysis and a costing analysis tool tasked with computing the electrical costs part of the simulation results. This project makes explicit something akin to architectural aspects of the design space by allowing the use of clusters, where each cluster may have different sets of parameters and constraints but all are ultimately compared using the same objective values.

3 INTO-CPS Case Study DSE Goals

In this section the case study owners present details of their scenarios from four distinct view points. These viewpoints consider the parameters that are present in each scenario, how each scenario will be measured, how the results of each designs will be compared to rank the designs and how the results could be presented. These aspects will affect the direction of the DSE module scripts over the next two years of the project.

3.1 Agro Intelligence

3.1.1 Design Parameters

In the agricultural case study we have two categories of parameters. The first category defines the robots physical size and its operation conditions. The second category defines the parameters of the surrounding environments. The first category has internal dependencies, like battery type/total weight and wheel size/operation speed. But there are also dependencies between the two categories, the wheel slip will affect the operating speed and the surface type will affect the wheel slip. Crop type will affect the width of the robot, because the robot needs to fit to the row distance for the current crop.

Category 1

- Wheel size, diameter in mm [450, 700].
- Width center of wheel to center of wheel, length in m [1.2, 3.5].

[10] `http://www.merlin-rail.eu`

- Battery type, [GEL, AGM, Lithium].
- Batery Capacity, [60, 150] Ah.
- Battery Weight, in kg [5, 20] per cell.
- Motor Power, [5, 25] kW.
- Motor Torque, [100, 35000] $\frac{mN}{m}$ (milli Newton)
- Gearing between motor and wheels, [10, 100].
- Total weight, [300, 1000] kg.
- Operating speed [0.3, 3] $\frac{m}{s}$.
- Component list.
- Component price.
- Implement type, [Seeder, Row-Crop Cleaner, Robovator, K.U.L.T. Duo,...]
- Implement Drag force N

Category 2

- Wheel slip on the surface [-100, 100] % (positive slip means the wheels are spinning and negative that they are skidding)
- Surface type [Bare soil, grass, pavement,...]
- Rolling friction coefficient between the surface and the wheel, has to be defined for each surface type.
- Crop type, the type defines the row distance. [100, 1000] mm.

3.1.2 Solution Objectives

The simulation goal for the agro-case is to determine the optimal vehicle configuration for a given scenario. The scenarios are defined by the implement, crop and surface type. The parameters that should be optimised for are: operation time, navigation such as turning radius and route plan in relation to the operation time. The end result should be a list of configuration parameters that can be used in the final vehicle design.

3.1.3 Ranking of solutions

Like in the automotive case 3.2.3, the different designs will be compared by a cost function. The parameters of the cost function will be total cost and maximum operation time of the robot in the given scenario and configuration.

3.1.4 Results Presentation

The result of the simulation should be presented in an interactive manner, where the user can select the parameter he/she wants to see the results for. It should be possible to select several parameters, e.g. battery capacity and wheel size. Along with the parameters it should be possible to define the scenarios the user wants to compare, e.g. what is the optimal battery capacity and wheel size for the surface type and this crop type. The result should be presented in a list and a graph so the user can see the how the result compare to each other. It should be possible to select the indivual simulations to see a detailed description of the scenario.

3.2 TWT Gmbh

3.2.1 Design Parameters

For the automotive case study, two categories of parameters can be differentiated: The first group of design parameters that can be varied during an DSE experiment defines the vehicle: vehicle mass, aerodynamic drag coefficient c_w, rolling friction coefficient c_{rr}, battery capacity C and the full load curve, defined by the maximum engine speed n_{max} and the maximum torque M_{max}. The second set of parameters defines the route the vehicle takes to get from the start position to its destination. These parameters can be described as a set of coordinates. For a typical DSE experiment in the context of INTO-CPS, the first set of parameters, defining the vehicle, is most likely the more relevant group.

The vehicle design parameters can depend on each other, e.g. the battery capacity has an influence on the total mass.

3.2.2 Solution Objectives

The simulation results that are of most interest for the automotive case study are relatively directly measurable. They include: the maximum acceleration should not be higher than a specific value (e.g. $4ms^{-2}$), the time that it takes to travel a certain distance, the time it takes to achieve a temperature inside the vehicle within the comfort zone ($T_{min} < T < T_{max}$).

3.2.3 Ranking of solutions

Different designs (i.e. vehicle configurations) are compared by using a cost-function that has parameters such as total vehicle cost, energy consumption, space and mass. In particular, electric vehicles are optimized using cost functions that include the battery capacity, energy consumption, mass, driving performance, efficiency per component and energy at the tire. Hybrid vehicles have cost functions that include the battery capacity, power of the elctric motor, power of the combustion motor, range, energy consumption, driving performance and efficiency of components. These cost functions are however individual for each automobile manufacturer and depend on the specific problem that needs to be solved. Therefore, there are no universal cost functions or rules for ranking of results.

3.2.4 Results Presentation

The range of electric vehicles is typically presented as a bar diagram for different vehicle configurations. For hybrid vehicles, the results could be shown in a 3D-plot, with the different working points of the combustion motor as the second parameter axis.

3.3 ClearSy

3.3.1 Design Parameters

For the railway case study, two kinds of parameters can be differenciated. The first group of parameters correspond to real numbers (or function of real numbers) such as Kinetic energy, communication or physical movement delay, track length, track slope (function of position) or traction acceleration (function of speed), or breaking force.

The other group of parameters is rather a choice of decomposition of a whole track map into several distributed one, and the corresponding distributed interlocking. Thus, such parameters are a set of subsets of the track map database tuples. On can also consider a varying number of trains.
The parameters may be related, such as minimal and maximal Kinetic energy or minimal or maximal slope or traction acceleration.

3.3.2 Solution Objectives

There are two kinds of measurement. The first kind of measurement are extremal values of monitored real number variables such as: train trip delay, Kinetic energy, train availability. The other kind of measurement is whether one train will overun another and collide, or whether two train collide because of an error in the interlocking PLCs. The two kinds may be dependent: one may want to measure availability but only in the case there is no collision.

3.3.3 Ranking of solutions

The preference value for solution objective may be the maximal or minimal value while such and such parameters vary (ex: track map distribution or traction vary and then the simulation tool would try to find the minimal train trip delay).

It is not clear what is the link between train trip delay and train availability. In this case, a curve with at least an extremal value (for instance the availability) in function of train trip delay could be fine (train trip delay computed with the other parameters the number of train, track map decomposition).

3.3.4 Results Presentation

The presentation for simple objective value would be a table or curve with a few significative simulations and showing the extremal objective value(s). For the case of a limit value is overun (such as maximal allowed speed, overun of train so positions overrunning, or collision with same), it could be interesting to show similar table or curve in the neighbourhood of this limit value. Finally, it should be interesting to show XY curve for showing trade

off between two competitive objectives. (such as availability vs train trip delay).

3.4 United Technologies

3.4.1 Design Parameters

DSE is used in building automation to: (a) identify the optimal equipment and control settings for an existing building; (b) study the equipment scalability over different building thermal characteristics. In the following we highlight the key design parameters used in the building automation case study:

- *Equipment Design Parameters:* tuning these parameters lead to identify the optimal thermal supply settings to a building using Fan Coil Units (FCUs).
  1. Maximum water flow rate: $\dot{m_{water}} \in [0.08 : 0.12]$
  2. Maximum air flow rate: $\dot{m_{air}} \in [0.4 : 0.6]$
  3. Water coil efficiency: $\epsilon_{coil} \in [0.1 : 1]$
- *Control Design Parameters:* tuning these parameters lead to identify the optimal PID control response to the building thermal load.
  1. Proportional set-point weighting: $K_p \in [0 : 1]$
  2. Derivative set-point weighting: $K_d \in [0 : 1]$
- *Plant Design Parameters:* these parameters are used to express different building heat dissipation characteristics. The building thermal parameters are varied based on ASHRAE fundamentals 2013 standard.
  1. Wall density: $\rho_{wall} \in [960 : 1600]$
  2. Wall thermal conductivity: $\lambda_{wall} \in [0.0865 : 0.1298]$

Considering that these parameters are independent, then the search space for optimizing equipment setting is $\dot{m_{water}} \times \dot{m_{air}} \times \epsilon_{coil} \times K_p \times K_d$. Whereas, the search space for equipment scalability study is $\rho_{wall} \times \lambda_{wall}$.

3.4.2 Solution Objectives

The objective of a building automation system is to maintain the user comfort, while minimizing the energy consumption. In our case study, the user comfort is represented as the room air temperature RAT, whereas more comfort metrics can be taken in account, such as CO_2 and humidity. The automation system maintains the RAT in the comfort band identified as $RAT_{sp} \pm 1^oC$, where RAT_{sp} is the RAT set-point identified by the user or the building manager. Therfore, evaluating these performance metrics requires observation of the following dependent variables:

1. Room Air Temperature RAT
2. Room Air Temperature Set-point RAT_{sp}
3. Supplied Power Q_{in}

3.4.3 Ranking of solutions

In order to rank the search space solutions, we formulate two evaluation metrics as follows:

- User discomfort UD is calculated as the area between RAT and RAT_{sp} curves. In order to minimize the user discomfort, the RAT needs to respect the RAT_{sp}. RMSE (Root Mean Square Error) is used to quantify the user discomfort as follows, where N is the total number of samples:

$$UD = \sqrt{\frac{\sum_{k=1}^{N} \left[RAT_{sp}(k) - RAT(k)\right]^2}{N}} \tag{1}$$

- Energy Consumption E is calculated as the integration of the used power Q_{in} over the time. In order to calculate the energy consumption, Coefficient Of Performance (COP) of the heat pump (HP) required to be considered as follows (i.e. COP=2.6), where T is the sample duration :

$$E = \sum_{k=1}^{N} \frac{Q_{in}(k) * T}{COP} \tag{2}$$

Optimizing the building performance requires minimizing both metrics $Min(UD, E)$. However decreasing one of them leads to increase the other, therefore a pareto frontier is required to be evaluated in order to identify the optimal design parameters.

3.4.4 Results Presentation

Considering the building automation is a multi-objective optimization problem, then we present the DSE results in an nD plot, where n is the number of the optimization criteria. In our case study, we optimize the building automation against two optimization criteria, i.e. E, UD. Therefore, the search space will be presented in a $2D$ plot that captures E and UD values for different configurations.

3.5 Key Points from the Case Studies

There are many key points from these descriptions that we can extract and take into account during the future development of the DSE model scripts.

- Each of the scenarios has a mixture of both design parameters of the system itself and parameters defining the environment in which the system is to operate. While these both contribute to the total design space to be explored it is important that we differentiate between them when, for example, grouping results by design.
- There are sometimes relations between simulation parameters meaning that not all combinations are valid, for example in the AI case study, the surface type of the ground affects the wheel slip parameter and in the TWT case study various parameters of the vehicles are linked, such as the battery capacity of an electric vehicle and its total mass. The parameter sweep should only visit parameter sets that respect these constraints.
- There is a range of different complexity levels when processing the raw simulation results to derive the objective measures needed to assess each simulation. Some are instantaneous measures that may be directly provided by the simulation outputs, such as the maximum acceleration of a vehicle, which others require more complex assessment. Examples include the time taken for the car cabin temperature to reach a comfortable level, the cumulative occupant comfort level in the UTRC scenario and computation of the turning radius in the AI study.
- There are also constraints over the variables in the simulations that must not be breached, an example of this is the detection of collision of two trains in the CLE case study. Such a constraint results in a boolean pass or fail that should be recorded amongst the objective results. It

may be advantageous to terminate a simulation when a constraint is breached to reduce wasted CPU time but this is outside the current planned capabilities of the DSE module.

- The UTRC case study explicitly calls for a pareto optimal type analysis to compare and rank the design results the TWT case study calls for cost functions that take into account multiple design parameters and simulation results and are unique to each vehicle simulated.
- In terms of presentation the case studies propose a range of visualisations from being able to select a range of graph types such as bar graphs, 2D and 3D plots. These plots could show a range parameter and results or focus in on interesting areas such as the neighbourhood around the maximum speed of a train. These are alongside the ability to compare any two values on an XY plot style and also pareto optimal style plots.

4 Outline DSE Approaches

This section presents an outline of the approaches to DSE that are planned to be implemented with the INTO-CPSproject. The section is divided into methods for searching the designs space (Sections 4.1 and 4.2) and methods for ranking the performance of the designs when there are multiple objectives (Section 4.3). During the text reference will be made to the 'DSE Driver' which could actually refer to either a human or a software agent, though in the case of this work the intention is that the role will be played as far as possible by software.

4.1 Single Parameter, Single Objective Search method

If we are optimising a single objective by varying only a single parameter and we are confident that the shape of the graph relating the parameter and the objective is uni-modal (only a single maxima or minima) then the Golden section search method may be employed. In this method it is necessary to perform three initial simulations s1, s2, and s3, with the position of s2 between s1 and s3 being according to the golden ratio, and for each obtain the objective value. The positions of s1, s2 and s3 here refer to the value of the parameter being varied. The next step is to divide the largest section, s1–s2 or s2–s3 and perform a new simulation, s4, again positioned according

to the golden ratio and obtain the objective value for s4. Depending on the relative objective values of s2 and s4 we can determine in which region the maxima/minima exists and thus repeat the process using either s1, s2 and s4 as the new triple of simulations or s2, s4 and s3. This process continues until the width of the section being searched is deemed to be small enough.

The DSE driver needs to know which parameter is being varied and the initial bounds. Being a closed loop method it requires access to either a single objective value or the output of a ranking function.

4.2 Multi Parameter Search Methods

The multi parameter search methods outline below all focus on the parameter search part of DSE and do not describe how to assess the objective values of the CPS, they do all assume that we are able to compare designs to give them at least an order of preference.

4.2.1 Exhaustive Search

The simplest form of multi parameter search, and the one that was implemented in the Crescendo tool, involves performing a complete search of all possible permutations of the parameters. This "brute force" approach is only applicable for small design spaces where the organisation has enough processor time available to run all simulations. The advantage of the method, is that it will always find the globally optimum results.

Here the DSE driver launches co-simulations with all combinations of the design parameters and does not require any feedback from the simulations.

4.2.2 Space Culling

It is possible for some systems, to cull areas of the parameter space based upon past results and in doing so avoid running simulations that are predicted to fail some criteria. For example, if the objective of the CPS is for a vehicle to be able to follow a track and at speed X1 the vehicle has already experienced understeering and so failed to follow the track, then there is little point in performing a simulation at Speed X1 + 10 as it is unlikely that it will be able to follow the track either. Thus we may cull the design space above X1.

To be able to use this method requires that the engineers are confident that there objective trends continue and so ...

Here the DSE driver would need to know the range of parameter values to be explored, the resolution for the initial search and the criteria to end the search.

4.2.3 Orthogonal Matrices

Orthogonal Matrices, also referred to as Taguchi Tables, is a method for assessing the effect of parameter variables on objective values without having to explore the entire design space.

Key to the concept of orthogonal matrices is the idea of parameters interactions. The idea is that the performance of a design is very likely not dependant on the values of individual parameter but actually is dependant on the set of all values acting in concert. To see the results of all interactions it is necessary to simulate every permutation of design parameters, which is identical to performing an exhaustive search described previously. At the other end of the interaction scale it is possible to ignore the effects of parameters interactions and performing sufficient simulations so that each parameter value is simulated at least once. An orthogonal matrix then is a compromise between an exhaustive search and only simulating with each parameter value only once, and the compromise is in terms of how many parameters are included in each interaction, larger numbers of parameters included in each interaction means more simulations need to be performed to see their effects and conversely small numbers of parameters included in each interaction requires fewer simulations. An orthogonal matrix defines, for a given number of parameters each with a set of possible values and for a defined interaction size (minimum 2), exactly what simulations need to be performed so that each interaction is simulated at least once. For example, if a design has three parameters, A, B and C each with their own set of values, then a matrix considering interactions of size 2 would include all combinations of the parameters A and B and also also combinations of parameters of A and C.

The results are analysed as a set of graphs, Fig. 1, where each graph plots the objective value against the values of a particular parameter or group of parameters, and from these graphs it is possible to estimate the best values or range of values for each parameter, then the simulations confirming the predictions of the graphs may be performed. Key to building an orthogonal

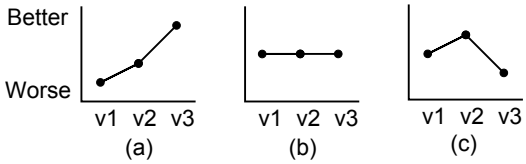


Figure 1: Simple example graph results from the Orthogonal Matrices method.

matrix is specifying the order of interactions that are to be considered. The smallest orthogonal matrices only consider the interactions of pairs of parameters and the largest matrix would consider all interactions, though this would produce a table that is identical to an exhaustive search.

Here the DSE driver would need to know the range of parameter values, the resolution of each parameter and also order of interactions to consider. It is open loop and so does not require simulation feedback.

4.2.4 Hill Climbing/Descent

Hill Climbing is a method where a point in the design space is chosen, either randomly or according to some engineering intuition, and then then the space is searched by repeatedly sampling (simulating) a neighbouring point in the design space and testing whether this point has a higher value than the current point and jumping to it if it does. This process continues until there are no neighbouring points with higher values and then the current design is returned as the maximum. Hill Descent follows the same procedure but selects neighbours with lower values instead. Both of these methods can find a maximum with fewer simulations than either a random or exhaustive search, but they have a weakness that in a system which contains both local and global maxima, they will not always find the global maxima.

There is also the steepest ascent/descent versions of this method, where instead of picking a single neighbour and determining if this is higher/lower, all surrounding neighbour designs are examined and the one with the highest/lowest value is chosen. This has the potential to arrive at a maxima/minima earlier than the normal method, but still suffers from the same weakness of becoming trapped.

Here the DSE driver needs to know the range of parameters and the resolution of each. This is a closed loop method and so it requires that the simulation results in either a single value or a comparable set of results.

4.2.5 Simulated Annealing

Simulated Annealing is related to Hill Climbing, but it attempts to avoid being trapped by local maxima by probabilistically being able to jump to a neighbour with a lower value if one is found. The probability of this jump to a seemingly worse design is initially quite high (>0.5) but as the number of simulations performed increases, the probability gradually decreases. The optimum design is either the one being examined at the end of the experiment when all simulations have been run or possibly an earlier one if the records of all designs are kept and there were some jumps to lower values near the end of the experiment.

The DSE driver needs to know the allowed values for the design parameters, the rate at which the probability of jumping decreases along with the total number of simulations to perform, and access to the simulation results.

4.2.6 Genetic Algorithms

The genetic method for exploring a design space takes its inspiration from the natural process of genetics and evolution. In its simplest form it starts by generating a number of random designs and each one is simulated to obtain its objective values. The designs are then ranked according to the results and the process of building the next generation begins by choosing a pair of designs to act as as parents. Two new designs are generated by combining the parameters values of the two parents such that the two parents parameter values are shared between the two offspring, with a small chance the one or more of those parameters will be mutated to have a different value. The new designs are tested via simulated to obtain their objective values and then the process starts again.

The DSE driver here needs the parameters and their allowed values, it is a closed loop method and so requires the simulation results and it also needs various tuning parameters such as the probability of parameter mutation and the number of elite individuals to maintain.

4.2.7 User Defined

This is not a DSE algorithm but instead it is simply some facility to allow the users to enter the parameters defining a series of simulations that will then be automatically executed in an open loop fashion.

$$V_a = w_1^a v_1^a(x_1^a) + w_2^a v_2^a(x_2^a) + ... + w_n^a v_n^a(x_n^a)$$

Figure 2: An outline WAM equation

The driver needs a value for each parameter for each simulation to be run.

4.3 Multi Objective methods

For those simulation models that have more than a single objective value and one of the closed loop DSE methods has been employed then it is necessary to use one of the objective methods to provide a means to perform any tradeoff between the individual objectives and determine which are the best designs. There are two types of method presented here, the two ranking methods order the design results by their ability to meet a specific goal, while the Pareto method will present a range of optimal tradeoffs. While each of the methods is quite different in how they perform their required computations, they all make the same assumption that the simulations can provide a single value for each objective.

4.3.1 Ranking Functions - WAM

The weighted additive method (WAM) computes a score for each design by multiplying the value of each objective by a weight for that objective and then summing the results. Figure 2 shows the basic form of a WAM equation.

4.3.2 Ranking Functions - ENUM

The enumerate and scoring method (ENUM) method computes a score for each design by defining groups of logical statements over the simulation results and then considering which and how many of those logical statements are true for the results of each simulation. Each statement within a group earns the same score if it is met, and the scores for each group are computed such that meeting a single statement in one group earns a score that is greater than meeting all of the statements in all groups below. Thus the engineer may uses the groups to express a preference for some conditions over others. The final score for a design is the sum of scores earned for all conditions met. The basic form of an ENUM equation is shown in Fig. 3.

$order_a$
{
$s_1{}^a = \{ g_1{}^1 : g_2{}^1 : ..., g_n{}^1 \};$ $score\_s_1{}^a = w;$
$s_2{}^a = \{ g_1{}^2 : g_2{}^2 : ..., g_n{}^2 \};$ $score\_s_2{}^a = y;$
s ...
$s_n{}^a = \{ g_1{}^n : g_2{}^n : ..., g_3{}^n \};$ $score\_s_n{}^a = z;$
}

Figure 3: An outline ENUM equation

4.3.3 Pareto Optimality

Pareto optimality is an approach to ranking that in which the engineer only has to state which objectives are of interest and whether they should be maximised or minimised. The searches for all results that meet the conditions for Pareto optimality, which are that, for each objective, it is not possible to find a design that improves on the value of that objective without degrading the value of another. When all designs that meet this criteria have been found, they are collectively termed the non-dominated set and all designs it contains are the best tradeoffs found so far for the selected criteria.

5 DSE Module Design and Progress

5.1 Outline of the DSE Module

The DSE module, when completed, will be composed of five different script classes, each playing their own part in a DSE run:

DSE Driver this is the main script in the DSE process, it contains the algorithm used to search the design space and as such is responsible for picking the simulation values to start with, which parameters to exercise through simulation each time after that and when optimisation is complete and therefore when to end the search. It orchestrates the execution of the other scripts.

COE Handler this script takes a simulation configuration stored in a specified directory and executes the simulation it represents by calling the COE. It terminates after it has written the raw simulation results in `results.csv` into the specified directory.

Objective Evaluation the purpose of this script is to analyse the raw simulation results in a specified directory to determine the objective values from those results. For example it might simply be looking for the maximum value for the power drawn by a motor, or it could be more complex using multiple results to calculate the comfort value as described in the UTRC case study earlier (Section 3.4.2). These results will be saved in `objectives.json` in the specified directory.

Ranking the ranking script has the job of comparing the objective values calculated for each sim to produce a global ranking of the designs. This ranking could be either via a ranking function or according to a pareto type analysis. These results are saved into a common `ranked.json` file in the DSE root directory.

Presentation the final script has the job of compiling all the numerical results in and or all of the `results.csv`, `objtectives.json` and ranked.json into the graphs and tables as required for the user to consume.

The orchestration of the scripts is shown in Figure 4 and here we see that it is not the case that each script is executed once, there are three nested loops:

DSE loop this is the main loop of the DSE and it continues until either the DSE algorithm determines that an optimal design has been found or until the entire design space has been explored.

Simulation loop the number of times this loop executes before exiting is determined by the DSE algorithm and how many simulations it needs to perform in between computing the global rankings. A second factor in determining how many times this executes is the availability of running COE instances which may allow parallel simulations to be executed if the DSE algorithm supports it.

Evaluation loop if the standard evaluation scripts are used then this loop will execute once for each objective to be calculated. If the user builds their own evaluation script then it may only need a single execution to obtain all objectives.

There a three main assumption made by the DSE module scripts:

- Python 2.7 is installed
- the required `config.json` is in the same directory as the DSE scripts
- the `curl` application is available in the command line path. It is avail-

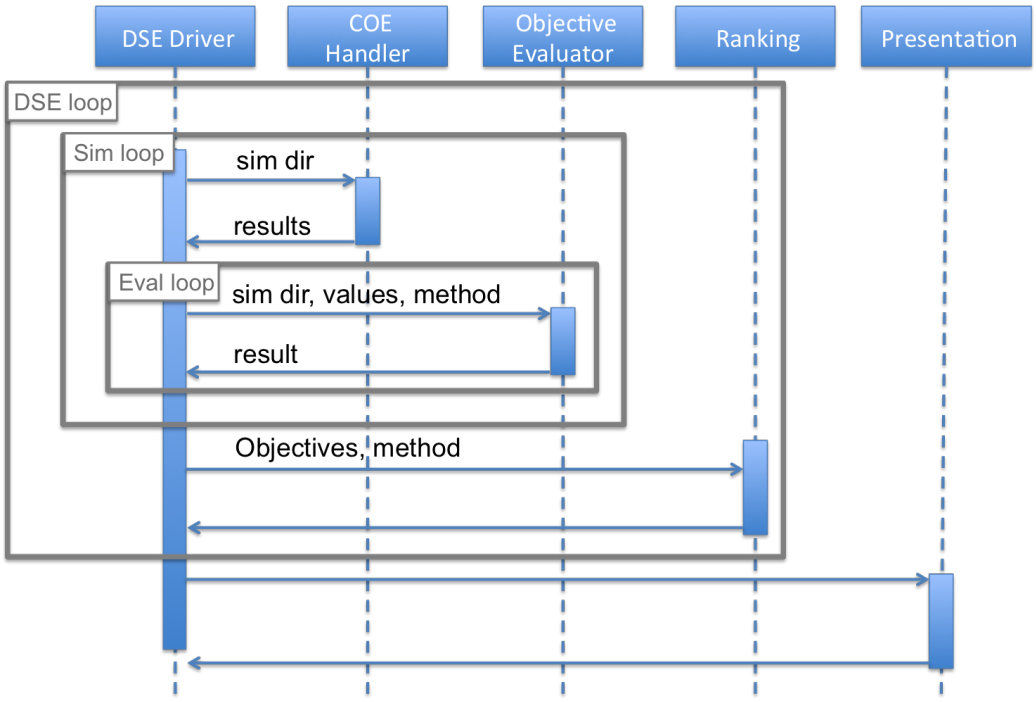


Figure 4: Informal sequence diagram of the DSE scripts

able by default in OSx and is downloadable for Windows[11]

Full instructions on the use of these scripts may be found in D4.1a [BLL+15]

5.2 Progress and Plans

5.2.1 DSE Driver

The first version of a DSE driver has been constructed. This script performs an exhaustive search, where all combinations of parameters are simulated in an open loop fashion. This script is included as Appendix A.1.

5.2.2 COE Handler

The first version of a COE handler has been constructed and tested. The script initialises, launches the COE and retrieves the results. It does not

[11] `http://curl.haxx.se`

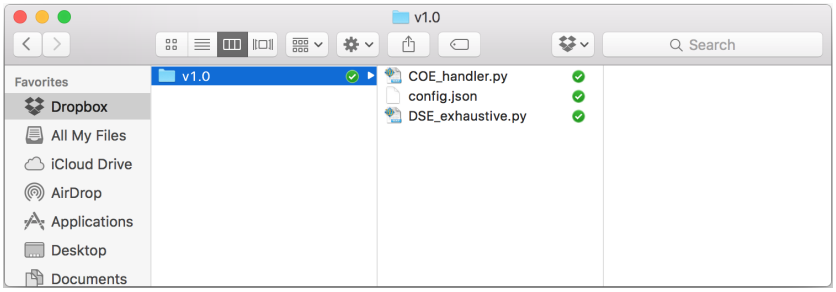


Figure 5: DSE initial folder contents

consider any faults at this time. The script is included as Appendix A.2

5.2.3 Objective Evaluation

A script allowing the evaluation of simple objectives, such a the maximum, minimum and mean values of any recorded variable is under development and may be ready for the final deliverable submission in December.

5.2.4 Result Ranking

The results ranking scripts have not yet been started

5.2.5 Result Presentation

The results presentation scripts have not yet been started.

5.2.6 Plan for development

A proposed ordering for the development of the scripts that comprise the DSE module is shown graphically in Figure 6. The plan consists of five columns, one for each of the script classes previously introduced, and the boxes on each line represent the method that will encoded in the a script and that point with time flowing from top to bottom. The ordering of the method developing is based upon a breadth first approach, preferring to have a set of scripts that spans from the driver through to the presentation of results before, for example, implementing all DSE search algorithms. It is hoped

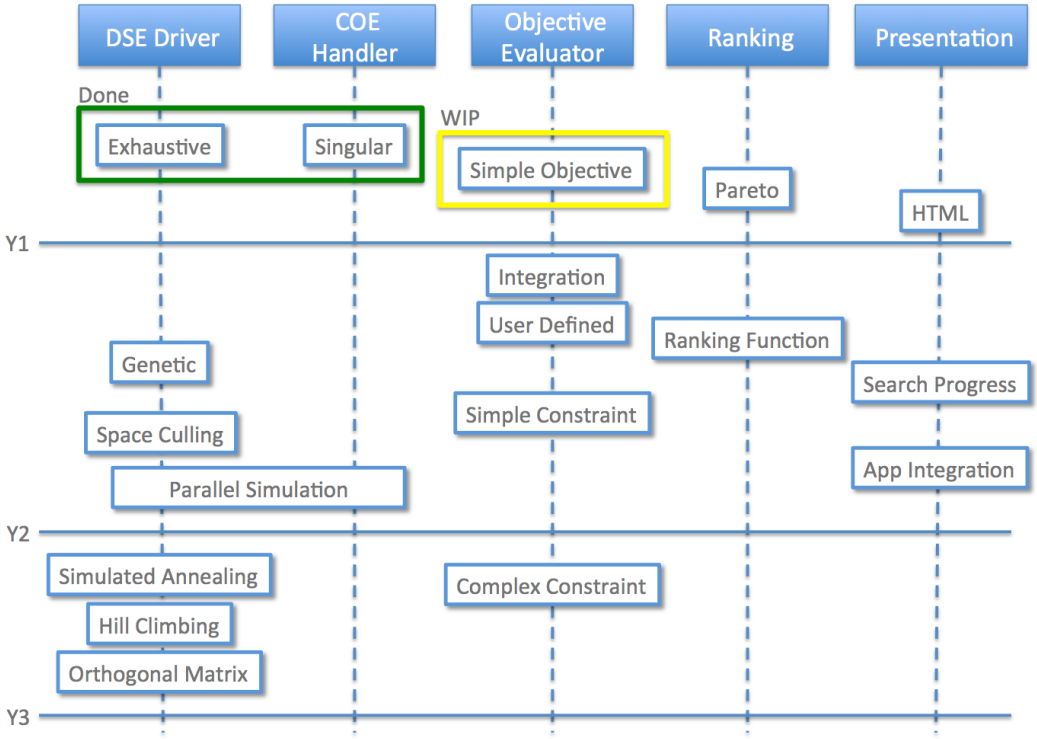


Figure 6: Proposed order for DSE script development

that this will best facilitate the adoption of the DSE module by the WP1 partners and therefore improve the flow of feedback.

The plan is split into three years, with each year ending with a blue line labelled. Y1, Y2 and Y3 respectively. In year 1 we see the completed scripts for exhaustive DSE search and the singular version of the COE handler have been completed (gree bounding box), we also see the simple objective script bounded in yellow, which is the current focus of our developments. Finally for this year we see that we intend to implement a Pareto ranking method and HTML presentation of the results. This will give us a complete, is computationally intensive, set of DSE methods.

In year 2, two more efficient DSE driver methods will be implemented, these are the genetic approach and space culling, along with further objective evaluation methods to both support the WP1 case studies and to allow the space culling approach to work. In terms of presentation, feedback on the search progress will be developed and importantly results feedback will be integrated with the INTO-CPSapp, along with the definition of DSE parameters. Since the DSE script is not integrated with the COE, as it effectively is in the

Crescendo tool, this opens up the possibility of running parallel simulations on multiple hosts which would have the effect of dramatically increasing the speed at which the design space could be explored. Such remote simulations could either be on spare machines or potentially using cloud based services.

In the final year, further DSE algorithms will be implemented along with more complex constraint checking and support for user defined ranking functions.

These time lines are conservative and it is hoped that items from year 3 may be brought forward to year, thus increasing the opportunity to respond to WP1 feedback and produce well founded guidelines.

References

[BFG+12] Jan F. Broenink, John Fitzgerald, Carl Gamble, Claire Ingram, Angelika Mader, Jelena Marincic, Yunyun Ni, Ken Pierce, and Xiaochen Zhang. Methodological guidelines 3. Technical report, The DESTECS Project (INFSO-ICT-248134), October 2012.

[BLL+15] Victor Bandur, Peter Gorm Larsen, Kenneth Lausdahl, Sune Wolff, Carl Gamble, Adrian Pop, Etienne Brosse, Jörg Brauer, Florian Lapschies, Marcel Groothuis, and Christian Kleijn. User Manual for the INTO-CPS Tool Chain. Technical report, INTO-CPS Deliverable, D4.1a, December 2015.

[CER13a] CERTAINTY. Modelling languages and models. Technical Report Deliverable D2.3, EU FP7 288175 CERTAINTY, 2013.

[CER13b] CERTAINTY. Preliminary methodology. Technical Report Deliverable D8.2, EU FP7 288175 CERTAINTY, 2013.

[FLPV13] John Fitzgerald, Peter Gorm Larsen, Ken Pierce, and Marcel Verhoef. A Formal Approach to Collaborative Modelling and Co-simulation for Embedded Systems. *Mathematical Structures in Computer Science*, 23(4):726–750, 2013.

[FNSV15] John B. Finn, Pierluigi Nuzzo, and Alberto Sangiovanni-Vincentelli. A mixed discrete-continuous optimization scheme for cyber-physical system architecture exploration. In *International Conf. Computer-Aided Design*, 2015.

[NBAR+12] Yunyun Ni, Jan F. Broenink, Kenneth G. Lausdahl Augusto Ribeiro, Frank Groen, Ken Pierce Marcel Groothuis, Carl Gamble, and Peter Gorm Larsen. Design space exploration tool support. Technical report, The DESTECS Project (INFSO-ICT-248134), December 2012.

A Scripts

A.1 DSE_exhaustive.py

```
import json, sys, os, platform, subprocess, time

print ("DSE Exhaustive launch v 1.0 starting")

#check OS
platform = platform.system()
if platform == 'Windows':
        print("Windows detected")
        pathSeparator = "\\"

if platform == 'Linux':
        print("Linux detected")
        pathSeparator = "/"

if platform == 'Darwin':
        print("Darwin detected")
        pathSeparator = "/"

runSimulations = True

def iterateOverParams(keyList, paramValues, simParamVals,
                                        start_time,  end_time ):
        keyListLocal = list(keyList)
        thisKey = keyListLocal.pop()

        for val in paramValues[thisKey]:
                simParamValsLocal = dict(simParamVals)
                simParamValsLocal[thisKey] = val

                if len(keyListLocal) >0:
                        iterateOverParams(keyListLocal,
                                          paramValues,
                                          simParamValsLocal,
                                          start_time, end_time)
                else:
                        outputSimParams(simParamValsLocal,
                                          start_time,
                                          end_time)
        return

def outputSimParams(simParamVals, start_time, end_time):

        newpath = makeDirName(simParamVals)
        print("    using params " + newpath)
```

```
        if not os.path.exists(newpath):
                os.makedirs(newpath)

        print("        results directory created")

        filepath = newpath + os.path.sep + 'config.json'

        configParams = parsed_json['parameters']
        for key in simParamVals.keys():
                configParams[key] = simParamVals[key]

        json_output = json.dumps(parsed_json, sort_keys=True,
                                 indent=4, separators=(',',': '))
        json_output_file = open(filepath,'w')

        json_output_file.write(json_output)
        json_output_file.close()
        print("        config created")
        #print simParamVals
        time.sleep(1)
        if runSimulations:
                launchSimulation(newpath, start_time, end_time)
        return

def makeDirName(simParamVals):
        first = True;
        dirName = ''
        for key in simParamVals.keys():
                if not first:
                        dirName += '-'
                dirName += str(simParamVals[key])
                first = False
        return dirName

def shortName(fullName):
        tokens = fullName.split("}")
        return tokens[1]

def launchSimulation(simFolder, start_time, end_time):
        subprocess.call(["python", "COE_handler.py", simFolder,
                                   start_time, end_time])

print ("Opening config.json")
json_data = open("config.json")
parsed_json = json.load(json_data)
params = parsed_json['parameters']


parameters = parsed_json['parameters']
```

```
paramvalues = {}

print ("Please enter a list of values for each parameter")

for key in parameters.keys():
        string_input = raw_input("    " + shortName(key) + ": " )
        paramvalues[key] = string_input.split( )


print("Please enter the start and stop times for the simulation")

start_time = raw_input("    Start: ");
end_time = raw_input("    End: ");


print("Starting the DSE...")
iterateOverParams(paramvalues.keys(), paramvalues, {},
                                start_time, end_time)

print("DSE complete.")
```

A.2 COE_handler.py

```
import sys, os, subprocess, platform, time, json

def getSessionKeyFromInitialisationResponse(rawInitResponse):
        parsedInitResponse = json.loads(rawInitResponse)[0]
        sessionKey = parsedInitResponse['sessionId']
        return sessionKey

platform = platform.system()

subdir = sys.argv[1]
start_time = sys.argv[2]
end_time = sys.argv[3]

print("        initiliasing simulation")

#initilise the coe and get session id
configPath = subdir + os.path.sep + 'config.json'
initialiseCmd = 'curl -s -H "Content-Type: application/json" \
       --data @' + configPath + ' http://localhost:8082/initialize'
initialisationResponse = subprocess.check_output(initialiseCmd,

sessionKey = getSessionKeyFromInitialisationResponse \
                      (initialisationResponse)

time.sleep(1)
```

```
print("        launching simulation")

startTime=0.0
endTime=5.0
runSimulationCmd = 'curl -s -H "Content-Type: application/json" \
             --data \'{"startTime":' + start_time + ', "endTime":'
\
             + end_time +'}\' http://localhost:8082/simulate/' \
             + str(sessionKey)
runSimulationResponse = subprocess.check_output(runSimulationCmd,

#print runSimulationResponse

time.sleep(1)

print("        fetching results")
getResultsCmd = 'curl -s http://localhost:8082/result/' \
                          + str(sessionKey)
getResultsResponse = subprocess.check_output(getResultsCmd,

resultsFile = open(subdir + os.path.sep + 'results.csv','w')
resultsFile.write(getResultsResponse)
```

B List of Acronyms

AU	Aarhus University
CLE	ClearSy
CLP	Controllab Products B.V.
DSE	Design Space Exploration
ENUM	Enumeration and Scoring
PROV-N	The Provenance Notation
ST	Softeam
TWT	TWT GmbH Science & Innovation
UNEW	University of Newcastle upon Tyne
UTRC	United Technology Research Center
UY	University of York
VSI	Verified Systems International
WAM	Weighted Additive Method
WP	Work Package