



Using Business Rules integrated with Optimisation for Complex Resource Allocation

Contents

abstract.....	3
Introduction.....	3
Description of the problem.....	4
Approach 1 to the problem.....	5
Stage 1 – Identifying and Creating the Problem.....	5
Stage 2 - Solving the Problem.....	8
Results.....	9
Approach 2 to the problem.....	11
Identifying and creating the problem.....	11
Results.....	11
Architecture of Intellify.....	11
Conclusions.....	12
References.....	13
Further Information.....	13

A TIGHTLY COUPLED INTEGRATION BETWEEN BUSINESS RULES AND CONSTRAINT PROGRAMMING USING INTELLIFY TO SOLVE A RESOURCE REALLOCATION PROBLEM

James Little and David Stynes
Thinksmart Technologies
james@thinksmarttechnologies.com

ABSTRACT

This paper describes a hybrid approach to solving a resource reallocation problem, where two technologies can work closely together to solve a problem in an elegant, efficient manner. By allowing the technology of Constraint Programming to be integrated through an Intellify middleware layer to Business Rules, increases the ease of harnessing the power of diverse, but complementary technologies. The problem being addressed is in dynamic machine utilisation, one which naturally breaks down into determining which machines within a factory exhibit performance difficulties and then reallocating the workload across a subset of these machines, to provide a better performance profile. The solving methodology allows us to use Business Rules technology to define the problem and to build an appropriate model for the Constraint Programming technology to solve for an optimal reassignment of work.

INTRODUCTION

There are a few instances of research in the combination of rules and constraints. Many of these papers [Kameshwaran et al, 2007; Carlsson et al, 2008; Fages and Martin 2008] relate to translating business rules into constraints and solving the problem with a single technology. This is not always applicable since business rules on their own can be used to make decisions using the different technology route of the RETE algorithm. Perhaps closest to the research presented here is the work by Bousonville et al [2005] who use Business Rules to generate the data input to a predefined optimisation model. In particular, they say, “we do not provide direct access to decision variables so that only predefined constraints are possible”. This is quite in contrast to what is demonstrated here through Intellify. We access the variables and indeed create the variable within the business rules environment. Rules are integral to solving the problem not as a pre-

Rules & Optimisation

processor or interesting way of managing data. LAURE [Caseau and Koppstein, 1992] is a system which allows the use of rules to guide the optimisation search. This is an example of close integration and a particular feature of the language, but one which could be configured easily, among others, within the Intellify architecture.

The automatic generation of dispatch rules (similar to business rules) for a packaging problem [van der Krogt and Little, 2008] through a CP optimisation model shows another hybrid configuration. Again though there is a loose coupling between the technologies. Here an optimisation model, reflecting the global demands over a time period, is used to choose the parameters for a set of rules describing which task and which machine to consider next.

Overall, there are a variety of valid ways (directions and levels) of combining rules and constraints which suggests that any platform technology needs to support the majority of these. In this paper we will demonstrate and contrast two hybrid approaches for solving the same problem.

DESCRIPTION OF THE PROBLEM

The problem is a real one, but is presented in a way which does not identify the company or the specific domain; instead it is described in general terms using common factory scheduling entities.

The 'factory' we are dealing with has a number of machines executing different tasks over time. The number of tasks on each machine is uncertain as the scheduling is done automatically using dispatch rules. It is felt that a balanced workload is desirable to prevent certain machines being overused and hence accelerating their required maintenance and downtime frequency. Sensors attached to the machines continually monitor characteristics of the machine which indicate how much "stress" the machine is under (e.g. temperature, rate of output, power usage, number of tasks, utilisation, etc). Consequently, some machines may become over used and at risk of breaking down. To address this problem, a set of rules, based on sensor information, is used to identify which machines are at risk. These rules identify situations, developing over time which should be conveyed to the operator as a classification of very high, high, medium and low risk machines. The operator then has to find a way of reallocating the work to try and alleviate the problem of machines at high risk. This is the problem we are concerned with in this paper. Although, there is no direct relationship between stress and utilisation it is a good enough indication and by moving tasks we change the utilisation and hence remove or increase "stress". There are in addition, certain restrictions of how the

Rules & Optimisation

workload can be redistributed. For certain machines there is a restricted set of other machines their workload can be transferred to. Overall we seek generate a re-distributed workload across all the machines with as few changes as possible.

APPROACH 1 TO THE PROBLEM

The approach naturally breaks into two stages:

1. The identification of machines which are at the different risk levels and the formulation of the re-distribution problem
2. An optimisation stage which generates a new workload allocation with as few changes as possible

Stage 1 – Identifying and Creating the Problem

Identifying the level of risk associated with each machine uses business rules (OpenRules www.openrules.com) to examine the sensor history of each machine. The machine risk classification rules are shown in figure 1, within an Excel table; the input medium for OpenRules. These rules describe the conditions for a machine to be at low (the default value, unless a rule raises it to a higher state), medium, high and very high state. For example, the first rule states that a medium risk machine is one which has spent 40% or more of its total time above a 45% utilisation level.

Rules void MachineUtilizationRules(Problem p, Machine m)							
C1	C2	C3		C4		A1	A2
frequency.compare(p.getTimeStep())	p.getMaxMachineUtilizationPercent(m) > utilPercentMax;	int timePercent = p.getTimePercentWithMachineUtilization(m,utilPercentMax); timeLimit.compare(timePercent)		int samples = p.getNumberofConsecutiveSamplesWithMachineUtilization(m,utilPercentMax); sampleLimit.compare(samples)		m.setExcessiveUtilizationAbove(utilPercentMax); String msg = "Machine " + m.getId() + " - " + warning + " " + utilPercentMax + "% of machine utilization"; p.addWarning(msg);	m.setUtilizationState(utilizationState);
CompareToInt frequency	int utilPercentMax	CompareToInt timeLimit	int utilPercentMax	CompareToInt sampleLimit	int utilPercentMax	String warning	int utilPercentMax
String utilizationState							
Sampling Frequency in minutes	Maximal Machine Utilization Percent	Time Utilization Percent	for Machine Utilization Percent Above	Number Of Consecutive samples	with Machine Utilization Percent Above	Produce Warning and Set Excessive Utilization Above Limit	
						Warning	Utilization Limit (%)
		>=40	45			:= "40% or more of time spent above of"	45
>=10	60					:= "At or above of"	60
<10	70					:= "At or above of"	70
<15				>=3	70	:= "3 or more consecutive samples above of"	70
>=15				>=2	70	:= "2 or more consecutive samples above of"	70
>=30				>=1	70	:= "1 or more consecutive samples above of"	70
							Set Machine Utilization State

Figure 1: Rule Description Interface for Classifying Risk

Rules & Optimisation

The third rule states that if there is a maximum utilisation at or above 70% when the utilization was being checked more frequently than every 10 minutes, then this machine is classified as high. Finally, the fifth rule states that if there are 2 or more consecutive utilisation readings above 70%, when the utilisation was measured at most every 15 minutes, then it is classified as very high risk. Since there are possibly multiple firing rules, the rules are triggered from top to bottom and each will override earlier rules if it is triggered. Thus the last rule that gets triggered will assign the machine the maximum risk level.

Intellify provides the facility to describe constraints within the Business Rules environment and to add them into the CP model. Therefore, we are able to describe through the same rules interface, the constraints on utilisation (figure 2). Here the constraint is to set the maximum utilisation to be either 60 or 70 depending on what the sampling frequency has been.

Rules void MachineUtilizationConstraints(Problem p, Machine m)		
C1	C2	A3
frequency.compare(p.getTimeStep())		m.setExcessiveUtilizationAbove(utilPercentMax); RelevantMachineRules (p,m,utilPercentMax);
CompareTo<int> frequency	int utilPercentMax	int utilPercentMax
Sampling Frequency in minutes	Reserved	Post Max Utilization Constraint
>=10		60
<10		70

Figure 2: Constraint Description Interface for the Maximum Utilisation

At the problem identification stage we can also start to build the features of the optimisation model in terms of the constraints and variables. We introduce a certain amount of intelligence in building of the model, since we believe that we should limit the number of machines to consider in the solution. We have to include all the machines which are at very high risk (or above the maximum utilisation level defined in figure 2), and we should also select those machines which are currently below a certain utilisation,

Rules & Optimisation

based on the historical maximum utilisation. We also ensure that for every very high risk machine with a set of optional machines for reallocation, and then a number of these are added in. Figure 3 describes the rules related to which machines to consider and Figure 4 shows the piece of Java code within OpenRules which adds the variable representing this machine to the model.

Rules void RelevantMachineRules(Problem p, Machine m, int limit)	
C1	A2
lowUsage.compare(p.getMaxMachineUtilizationPercent(m)) p.getMaxMachineUtilizationPercent(m) > limit	p.addConstraintMachineMax(m, limit);
CompareToInt lowUsage	int dummyValue
Highest Usage Machines to add to model	Post Max Utilization Constraint
<=15	0

Figure 3: Rule for Adding a Variable and Constraint for High and Low used Machines

```
/**
 * Adds a Var for the passed machine and the
 * Posts and returns a constraint limiting the maximum utilization of the machine
 * @param machine the machine whose maximum utilization is being limited
 * @param limit the limit on maximum utilization
 * @return the posted constraint
 */

public Constraint addConstraintMachineMax(Machine machine, int limit) {
    machineMaxUtilizationLimit = limit;
    Var var = csp.addVar(0,100,machine.getId());
    Constraint ct = var.le(limit);
    ct.setName(machine.getId() + " <= " + limit + "%" );
    ct.post();
    return ct;
}
```

Figure 4: Code Segment showing Rules passing information to the CP solver

Finally, rule and the compatible machine sets are shown in figure 5. These types of constraints make the problem significantly more difficult, whereas previously some instances of the problem could be solved by hand.

Rules void addCompatibleMachineSets(CompatibleMachineSets cms)					
Actions					
cms.addSet(compatibleSet);					
String[] compatibleSet					
Machine 1	Machine 2	Machine 3	Machine 4	Machine 5	Machine 6
XX-15A:D3	XX-15C:C2	XX-15D:D2	XX-16B:C0	XX-16C:D0	XX-01A:C0
XX-16C:D0	XX-01A:C4				

Figure 5: Machine Compatibility Sets

Stage 2 - Solving the Problem

The optimisation problem relates to finding a suitable redistribution of work which reduces the very high risk machines to a below a specified level, while affecting the minimum number of machines to accommodate this. Work we assume is simply a percentage and reflects the utilisation. Therefore, a machine having 50% utilisation is handling 50 units of work, some or all of which could be passed, in the same units, to another machine if this is desired. Here we do not model individual tasks, but assume that tasks can be subdivided across machines in discrete amounts of a single unit. The final decision as to what specific tasks are redistributed is left to the planner – this model simply indicates the feasibility and new allocation of work.

The model is built around a set of variables representing the utilisation of each machine. The value this variable can have is an integer between 0 and 100. The constraints are,

1. an upper bound on the utilisation of any machine and
2. a restriction as to which set of machines, one machine may redistribute its work.

The objective is to find a redistribution of work with the minimum number of changes to the workloads of the machines. The magnitude of the change is not considered as it is believed that the overhead of just making a change is more important than the quantity of the change.

The search chosen is based on first trying to find a solution with a lower bound of changes. The lower bound is taken as the number of machines at very high risk level plus 1 (in other words if the work on all the very high risk machines could be redistributed to a single machine). If unsuccessful, the search then increments this counter until a first solution is found. This by implication is the optimal solution.

With the number of machines totalling potentially over 100, the chances of finding and proving optimality are quite small. With the search strategy adopted a lot of searching will have to take place to prove infeasibility since there is a high degree of symmetry. Instead the approach we propose is that a smaller model can be generated, depending on the size and nature of the problem defined by the rules.

Our hypothesis is that a small problem will find better solutions within the solving time than a larger model, using the same search strategy.

As an aside a comparable technology we may consider using in place of CP is an Integer Programming Solver. This too is available using Intellify. However, the constraint of compatibility machines and the objective of minimum number of changes are not ones

Rules & Optimisation

which naturally linearises for use with IP. In this instance CP was chosen as the first technology option.

Results

The experiments show the different qualities of solution and time it takes to find these solutions.

Scenario 1

All machines are considered in the model and a solution is sought with a minimum number of changes. The model did not succeed in finding a solution even after one hour.

Therefore, we proceeded to incorporate the model reduction rule. The rule selected only those machines for inclusion into the model, which were at very high risk or currently below 15% utilisation.

In the subsequent scenarios, we show the 'before' and 'after' machine risk. The red machines are deemed very high risk, the yellow high risk, the green medium risk and the blue low risk.

Scenario 2

The problem has no objective, but is simply to find the first solution which satisfies all the constraints. The result is that 4 changes to non-excessive machines were made in less than 1 second.

Rules & Optimisation

XX-15A:C0	XX-15A:C2	XX-15A:C4	XX-15A:C6	XX-15A:D1	XX-15A:D3	XX-15A:D5	XX-15B:C1	XX-15B:C3	XX-15B:C5
XX-15B:D0	XX-15B:D2	XX-15B:D4	XX-15B:D6	XX-15C:C0	XX-15C:C2	XX-15C:C4	XX-15C:C6	XX-15C:D1	XX-15C:D3
XX-15C:D5	XX-15D:C1	XX-15D:C3	XX-15D:C5	XX-15D:D0	XX-15D:D2	XX-15D:D4	XX-15D:D6	XX-16A:C1	XX-16A:C3
XX-16A:C5	XX-16A:D0	XX-16A:D2	XX-16A:D4	XX-16A:D6	XX-16B:C0	XX-16B:C2	XX-16B:C4	XX-16B:C6	XX-16B:D1
XX-16B:D3	XX-16B:D5	XX-16C:C1	XX-16C:C3	XX-16C:C5	XX-16C:D0	XX-16C:D2	XX-16C:D4	XX-16C:D6	XX-16D:C0
XX-16D:C2	XX-16D:C4	XX-16D:D1	XX-16D:D3	XX-16D:D5	XX-01A:C0	XX-01A:C2	XX-01A:C4	XX-01A:C6	XX-01A:D1
XX-01A:D3	XX-01A:D5	XX-01B:C1	XX-01B:C3	XX-01B:C5	XX-01B:D0	XX-01B:D2	XX-01B:D4	XX-01B:D6	XX-01C:C0
XX-01C:C2	XX-01C:C4	XX-01C:C6	XX-01C:D1	XX-01C:D3	XX-01C:D5	XX-01D:C1	XX-01D:C3	XX-01D:C5	XX-01D:D0
XX-01D:D2	XX-01D:D4	XX-01D:D6	XX-02A:C1	XX-02A:C3	XX-02A:C5	XX-02A:D0	XX-02A:D2	XX-02A:D4	XX-02A:D6
XX-02B:C0	XX-02B:C2	XX-02B:C4	XX-02B:C6	XX-02B:D1	XX-02B:D3	XX-02B:D5	XX-02C:C1	XX-02C:C3	XX-02C:C5
XX-02C:D0	XX-02C:D2	XX-02C:D4	XX-02D:C0	XX-02D:C2	XX-02D:C4	XX-02D:C6	XX-02D:D1	XX-02D:D3	XX-02D:D5

Desired Solution Type:

Normal Read Current Utilization

Minimise Changes Desired Peak Usage

Find Solution

XX-15A:C0	XX-15A:C2	XX-15A:C4	XX-15A:C6	XX-15A:D1	XX-15A:D3	XX-15A:D5	XX-15B:C1	XX-15B:C3	XX-15B:C5
XX-15B:D0	XX-15B:D2	XX-15B:D4	XX-15B:D6	XX-15C:C0	XX-15C:C2	XX-15C:C4	XX-15C:C6	XX-15C:D1	XX-15C:D3
XX-15C:D5	XX-15D:C1	XX-15D:C3	XX-15D:C5	XX-15D:D0	XX-15D:D2	XX-15D:D4	XX-15D:D6	XX-16A:C1	XX-16A:C3
XX-16A:C5	XX-16A:D0	XX-16A:D2	XX-16A:D4	XX-16A:D6	XX-16B:C0	XX-16B:C2	XX-16B:C4	XX-16B:C6	XX-16B:D1
XX-16B:D3	XX-16B:D5	XX-16C:C1	XX-16C:C3	XX-16C:C5	XX-16C:D0	XX-16C:D2	XX-16C:D4	XX-16C:D6	XX-16D:C0
XX-16D:C2	XX-16D:C4	XX-16D:D1	XX-16D:D3	XX-16D:D5	XX-01A:C0	XX-01A:C2	XX-01A:C4	XX-01A:C6	XX-01A:D1
XX-01A:D3	XX-01A:D5	XX-01B:C1	XX-01B:C3	XX-01B:C5	XX-01B:D0	XX-01B:D2	XX-01B:D4	XX-01B:D6	XX-01C:C0
XX-01C:C2	XX-01C:C4	XX-01C:C6	XX-01C:D1	XX-01C:D3	XX-01C:D5	XX-01D:C1	XX-01D:C3	XX-01D:C5	XX-01D:D0
XX-01D:D2	XX-01D:D4	XX-01D:D6	XX-02A:C1	XX-02A:C3	XX-02A:C5	XX-02A:D0	XX-02A:D2	XX-02A:D4	XX-02A:D6
XX-02B:C0	XX-02B:C2	XX-02B:C4	XX-02B:C6	XX-02B:D1	XX-02B:D3	XX-02B:D5	XX-02C:C1	XX-02C:C3	XX-02C:C5
XX-02C:D0	XX-02C:D2	XX-02C:D4	XX-02D:C0	XX-02D:C2	XX-02D:C4	XX-02D:C6	XX-02D:D1	XX-02D:D3	XX-02D:D5

Scenario 3

The model has objective of finding the smallest number of changes. The result is that 3 changes were made to non-excessive machines. The solution time was about 1 second.

Scenario 4

The problem has two additional compatible set constraints. The result appears in the interface that only 2 new cells have changed colour, actually cells XX-16C:D0 and XX-01A:C0 have been increased too; they are just still low utilisation so the colour did not change. Thus 4 changes to non-excessive machines were made here. The running time was about 5 seconds. The increase in running time can be explained by the failure driven search we are using. It proceeds by trying to find a solution with one change and on failing proceeds to 2 etc. With the additional constraints we extend the search from 3 to 4 changes and so it takes longer.

These results show that we can firstly obtain solutions and indeed optimal solutions provided that we reduce the model size. Secondly, the quality is not compromised through model reduction; scenario 1 did eventually provide a solution of 3 changes. If

Rules & Optimisation

additional constraints are added, the time to find a solution increases while the quality of optimal solution goes down.

APPROACH 2 TO THE PROBLEM

The approach is described in two inter-leaving stages:

1. As each machine is sequentially identified as high risk using the rules definition, a small local problem is created and modelled, but within a restricted set of machines 'close to' to the high risk machine. This is then passed to stage 2
2. An optimisation stage which generates a new workload allocation with as few changes as possible.

This approach is particularly relevant where the presence of high risk machines is fairly sparse and there is the opportunity to create several small problems. This is a 'local search' approach where we try to improve the solution incrementally. Again we have the same type of constraints and objective function. The only difference is in the additional neighbourhood rules we define to describe what size of local problem to create.

Identifying and creating the problem

This follows much the same process as before. However, we use some new problem creation rules. These rules define the 'distance' from the high risk machine that we are considering. If there is any overlap of machine sets in the problems, the updated values from the first local search are used as the starting values in the second problem model.

Results

On scenario 1 data set, there are 8 excessive machines, and this approach results in 8 local problems being solved, each making 1 change and resulting in 8 non-excessive machine changes overall.

ARCHITECTURE OF INTELLIFY

Bringing rule and constraint solving engines together has been achieved through the Intellify middleware, the full architecture of which is shown in figure 5. Intellify also provides adaptors to a number of constraint solvers so that a model defined in Intellify's modelling language can use a variety of solvers depending on the requirements of performance and licensing.

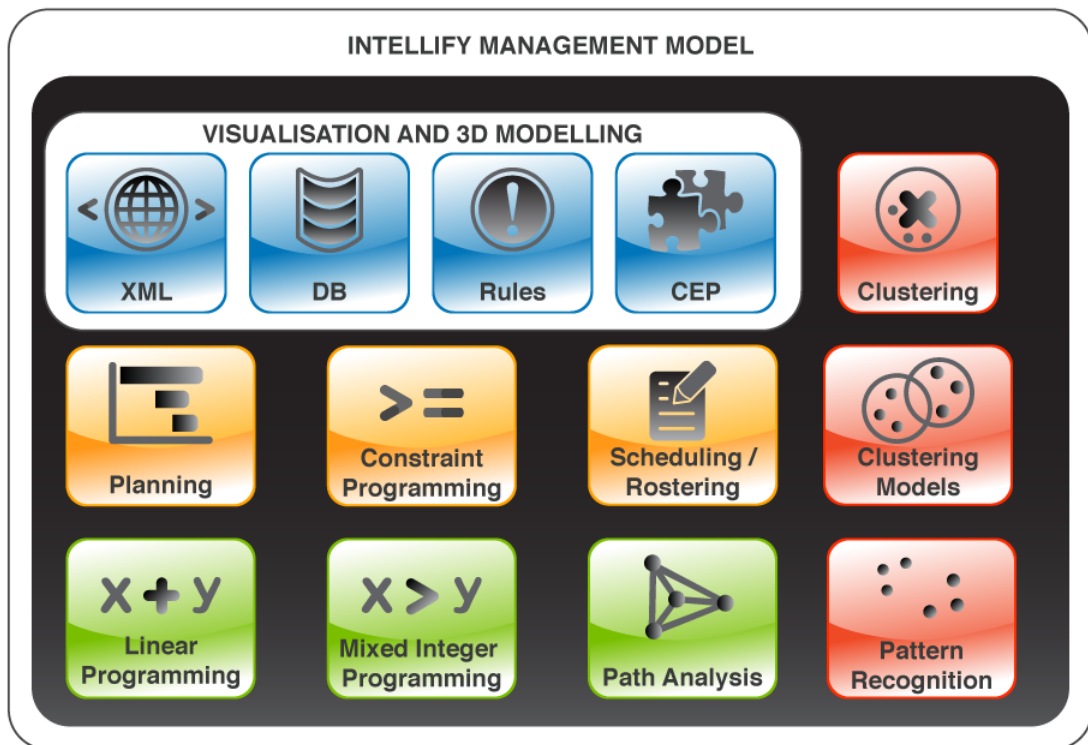


Figure 5: Application Architecture of Intellify

CONCLUSIONS

The integration of rules and constraint based optimisation technologies through Intellify creates a platform which allows the development of a variety of solving configurations. These configurations are capable of solving complex problems which require a mixture of approaches. Making these approaches fit together easily is the first requirement for a successful decision making technology. Intellify provides the middleware to bring Constraint Programming into different business tools environments and be accessed easily from there. The application discussed here demonstrates the need for taking an integrated approach to solving a complex problem, which can be solved simply by appropriate decomposition. The problem in question is illustrative; more complex problems with additional side constraints and more complex risk conditions would justify the approach more. For Business Rules finding an optimal solution is problematic and could conceivably lead to many rules which are difficult to maintain. For Constraint Programming the creation of the instance of the problem model would require much programming. By passing each challenge to a different technology and linking them together at a low level delivers a fast, easily maintainable solution. Finally, for Business

Rules & Optimisation

Rules users, this gives them access to an optimisation technology without leaving their natural problem description environment.

REFERENCES

T. Bousonville, F. Focacci, C. Le Pape, W. Nuijten, F. Paulin, J-F. Puget, A. Robert and A. Sadeghin, "Integration of Rules and Optimization in Plant PowerOps", Principles and Practice of Constraint Programming, Lecture Notes in Computer Science Springer Volume 3524, pp 1-15, 2005.

M. Carlsson, N. Beldiceanu, and J. Martin, "A Geometric Constraint over k-Dimensional Objects and Shapes Subject to Business Rules", Principles and Practice of Constraint Programming, Lecture Notes in Computer Science Springer Volume 5202, pp 220-234, 2008.

Y. Caseau and P. Koppstein, "A cooperative-architecture expert system for solving large time/travel assignment problems", In Database and Expert Systems Applications, pp 197–202, Valencia, Spain, 1992.

F. Fages and J. Martin, "From Rules to Constraint Programs with the Rules2CP Modelling Language", INRIA Research Report RR-6495, Institut National de Recherche en Informatique (2008).

S. Kameshwaran, Y. Narahari, C. H. Rosa D. M. Kulkarni and J. D. Tew, "Multiattribute electronic procurement using goal programming", European Journal of Operational Research, 179, 518–536, 2007.

R. van der Krogt and J. Little, "Optimising Machine Selection Rules for Sequence Dependent Setups with an Application to Cartoning" in Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM-09), 2009.

FURTHER INFORMATION

Thinksmart Technologies Limited

Innovation centre, Western gateway,

UCC, Western road, Cork, Ireland

www.thinksmarttechnologies.com