Optimal Requirements-Dependent Model-Driven Agent Development

Joshua Z. Goncalves *joshua.goncalves@curtin.student.edu.au*
Curtin University / Department of Computing
Perth / WA-6102,Australia

Aneesh Krishna *A.Krishna@curtin.edu.au*
Curtin University / Department of Computing
Perth / WA-6102,Australia

Abstract

The Belief-Desire-Intention (BDI) agent model is a highly favoured agent development model known for its distinct abstraction between components, conceptual adaptability and flexibility in determining its actions. This determination is handled through a plan selection function which determines the most appropriate plan or action given the current state of the environment. Whilst it is conceptually easy to understand, the BDI platform remains guarded by a particularly steep learning curve, especially with regards to implementation and any required adaptation. Recent years have seen various forms of extensions and approaches to BDI agent models, including a model-driven approach based around the Extended Non-functional requirements framework. Non-functional requirements illustrate parts of a system which must be satisfied to an appropriate extent. These requirements remain indeterministic in their nature however, such that their satisfaction cannot be done directly. The model-driven approach within this paper uses components from this framework to formulate plans which are governed by their contribution to these requirements. This is done in an optimised manner to ensure the selected plan is optimal in regards to the systems overall attainment. To our knowledge, this is the first time an optimised approach has been used in relation to model-driven agent creation. This paper presents our optimised model-driven agent development approach, demonstrating its conversion from the initial extended non-functional requirements model into a completely optimised and functional agent. The approach is verified through experimental analysis.

1. Introduction

Autonomous agents are steadily becoming a viable solution for various applications in the industry due to their high level of adaptability and flexibility. This makes them a more appropriate alternative to traditional software implementations depending on the context [3]. Creating these agents however has long been a difficult feat in modern software engineering, with their implementation and approach being very application-specific and non-transferable. This is mainly due to the expertise required both in the actual implementation of the agents and the methodology/design approach required to create them successfully.

With the rise in popularity of the Belief-Desire-Intention (BDI) model [11], this agent creation process has become conceptually easier and technically more capable. Currently, there exists a multitude of techniques [13, 5] which rely and build upon it as well as accompanying platforms [7, 10] which implement it. Regardless of this however, the entry-level knowledge requirement is still high, with many developers understandably choosing to stick with a comfortable alternative as opposed to the economical risk. In sight of this, model-driven agent development (MDD) [6] frameworks (such as the one in this paper) are being explored to reduce the agent-orientated aspects to the background. This means that the developer only needs to focus on the traditional programming aspects.

Our approach is based on the Extended Non-Functional Requirements (ENFR) framework [1, 4] which uses the original NFR framework but replaces the qualitative reasoning to instigate a quantitative approach allowing determinism. Our approach uses the preferences given to operationalisations within the ENFR model as well as the progressive values of its children to determine which operationalisation is the most appropriate. This means that the selected operationalisation is optimal both by itself and through the potential path it can take through its children. Using this approach, based on any combination of operationalisations available at a particular moment, the optimal path with respect to both time and space complexity will be followed.

In the next section (Section 2), we provide background knowledge on both the BDI agent framework and the Extended NFR-framework which the model is based on. In Section 3 we provide an explanation of the development and methodology surrounding our optimised model-driven agent creation approach. In Section 4 we present our optimisation model mathematically and how it functions. In Section 5 we evaluate our optimised approach empirically against static, random and preferential-based approaches. In Section 6 we present work related to our approach and finally, in section 7, we conclude the paper.

2. Background

This section provides details on background concepts and methodologies which are required in order to understand subsequent sections of this paper.

2.1. Belief-Desire-Intention Agent Framework

The BDI Agent framework specifies a certain structure which enables agent creation in a conceptually adept, flexible and powerful manner. A practical agent implemented using this framework is basically a dynamic planning system that determines which plans to execute in order to achieve its goals [11]. BDI agents are conceptually split into three different parts, as denoted by their acronym. *Beliefs* represent information from the agent itself, its environment, and other agents that it communicates with. Internally, these beliefs are represented as variables with their value denoting the state of the belief. The agent believes this information to always be true.

Desires are objectives which the agent strives to achieve, with the objectives representing specific states of the environment. These are represented as goals which can be triggered by both the agent itself and by perceptions received from the agents own environment.

Intentions represent a series of pre-defined actions which the agent has decided to take in order to make a chosen desire true. These are represented as plans or operations which the agent is capable of performing. In order to satisfy some desires, multiple plans following a sequence are sometimes necessary.

We now describe a typical execution cycle from a BDI agent following a default implementation. Initially the BDI agent will receive a percept (perception based on either its internal or environmental state) which will trigger a range of possible plans. The triggered plans will then have their various execution conditions checked to create a further subset of triggered, applicable plans. From this subset, a plan is chosen and executed. This plan selection strategy is known as the plan selection function which determines the plan that is most appropriate from the range of available plans given the agents context and percept sequence to date.

2.2. Extended NFR Framework

The extended NFR-framework revolves around two main entities, which are *softgoals* and *operationalisations*. Softgoals represent non-functional requirements, which are requirements that cannot be determinatively satisfied such as performance or safety. Each softgoal has a numerical weighting which determines its importance within the system. Operationalisations are ways of achieving these softgoals. For example, by achieving operationalisations with regards to the performance softgoal, it can be seen that the system will be sufficiently fast. Both of these components can be further decomposed into child variations.

Each apex operationalisation makes a contribution to a leaf softgoal which represents the operationalisations impact on that softgoal. These contributions are each given a value to denote their magnitude. This value, multiplied by the softgoals weighting, creates a summation which is assigned to the respective operationalisation as its score/preference. The operationalisations with positive scores are then selected to be implemented into the system, since this positivity means that they are beneficial. From the set of selected operationalisations, the attainment of the overall system can then be calculated. This is done by generating a summation of the contributions to the softgoals connected to the accepted operationalisations, multiplied by their weighting. This attainment score represents the degree at which all softgoals within the system have been satisfied.

Due to content constraints, this framework will not be explained in further detail, please refer to the accompanying paper [1] for more information if required.

3. Optimised Model-Driven Agent Development Approach

This section provides an explanation of the methodology surrounding our model-driven agent development approach. This begins by detailing an overview of the ENFR-to-agent mapping and how this is maintained (Section 3.1). The optimisation procedure is then detailed (Section 3.2) followed by the plan selection method which utilises it (Section 3.3) and ending with our customised ENFR-diagram format.

3.1. ENFR-to-Agent Mapping

From the original ENFR diagram, various components of it can be directly mapped to our agent. This starts with the belief state of the agent. The agents belief state consists of an abstracted version of the ENFR diagram, containing all of the components and relationships within it. This means that each operationalisation and softgoal gets their own data objects containing their relationships and values, including an identifier. These objects are arranged in a tree-structure within the system, providing quick traversal and access to each of them.

The mapping of plans within the agent starts with *System plans*, which are initial trigger points in the agent based on a pre-defined series of softgoals. Based on a series of percepts, a system plan will be executed. This plan will then create goals which will trigger the subsequent operationalisation plans. Each operationalisation within the agent gets their own plan, whose computation relates to the operationalisations purpose. These plans are divided into two types based on whether or not the operationalisation makes a contribution, known as an impact or basic operationalisation plans respectively. Within these plans exists the operationalisation object within the agents belief state, which is referenced at various points through out the agents progression. By including this, any changes to the belief state that relates to this operationalisation can be automatically incorporated by the plan. Softgoals within the agent are represented only by their object and are not mapped to an explicit BDI agent component, although they are referenced by the operationalisation plans themselves.

Using the above mapping, we now detail how a typical execution cycle occurs. The agent initially detects a series of percepts from its environment. Using these percepts it triggers a system plan. The system plan will then create a goal based on one or more softgoals and execute it within the agent. The operationalisation plan(s) contributing to these softgoals are then triggered by this goal, with the plan selection function executing the most beneficial one from

Mapping Rule	**Transformation**
Agent	Load ENFR Graph structure Initialise operationalisation scores Perform preliminary propagation
Plan	operation $\leftarrow$ Specific Operationalisation precondition $\leftarrow$ comparison value
Softgoal	Initialise Softgoal Object Set weighting and Identifier in new object
Operationalisation	Initialise Operationalisation Object Set probability and Identifier in new object Initialise operationalisation parent, children and softgoal links

Table 1. Mapping Rules

this set. Once this plan has finished its proprietary computation, it will reference the agent to trigger its children, with respect to the relationship that it has with them. An abstract version of this mapping can be seen in Table 1.

3.2. Optimisation Preparation

When deciding between a series of operationalisation plans, the agent ideally wants to select the plan which will achieve the highest level of attainment for the overall system. This is in relation not just to the operationalisation itself, but also to any children which it may have and the optimal selection amongst them. This is because the parent operationalisation is dependent on the completion of its children. Therefore, if the children's completion probability is low, or they contribute to additional softgoals within the system, then the overall benefit achieved by their parent will be effected. In order to properly handle this, our optimisation procedure adds two additional values to each operationalisation. The first is the *Preview* value which is a percentage that represents the degree of change between the operationalisations preference and the preference of its children. The second value is the *Preview Depth* which represents the number of child operationalisations participating in the optimal path, including itself. By multiplying the preview value against the preference of an operationalisation, we reach the *Comparison Value*. This value represents the degree of benefit from the most optimal path taken from the current point.

Creating the preview value is done through two forms of propagation, depending on the situation. The first is the preliminary propagation which occurs once the agent has been created. For each apex operationalisation, its preview value is created by propagating downwards to its leaf-child operationalisation. Once this child has been reached, its preview value and preview depth are calculated, which, considering that it has no children, will both be one. It then moves upwards to its parent and calculates their preview value. This is done by, for each of the parents children depending on their relationship, calculating their original average by multiplying their preview value by their own preference value. A summation for all of these child averages is then created, with the parent's preference value being added to it as well. A summation for the children's preview depths is created alongside this, being incremented to take the parent into account. The summation of the child values is then divided by the summation of the preview depth and then further divided by the parents original preference to create its preview value. This continues for every parent until the original apex operationalisation has been reached. The second form of propagation is from an operationalisation upwards through its parents. This is to increase efficiency, since if this operationalisation is modified, then the only preview values affected are its own and its parents. This process is the same as the initial propagation method.

Within both propagations, the relationship shared between the child and parent must be observed, which is either inclusive (AND) or exclusive (OR). Within an inclusive relationship, the preview values of all children must be taken into account. This is because, when initialising the parents preview value, each child must be executed in order to complete the parent. For exclusive relationships, only the child with the largest comparison value needs to be considered, because it is clear that the most optimal path through the system occurs through this child. This propagation process can be seen in Algorithm 1, denoting both the inclusive and exclusive preview scenarios.

Algorithm 1 PreviewPropagation(Operationalisation operation)

```
Input: Operationalisation whose preview value needs to be determined/updated
1:  $preview,\ previewDepth,\ tempValue\ =\ 0.0$
2:  for each Children in Operation do
3:      if child.comparisonValue $>$ tempValue AND relationship $=$ OR then
4:          previewDepth = $child.previewDepth$
5:          preview = $child.comparisonValue \cdot child.previewDepth$
6:          tempValue = child.comparisonValue
7:      else
8:          tempValue = $child.comparisonValue \cdot child.previewDepth$
9:          preview += tempValue
10:         previewDepth += $child.previewDepth$
11:     end if
12: end for
13: preview = $(preview + operation.preference)/(previewDepth + 1)$
14: operation.previewValue = preview / operation.preference
15: operation.previewDepth = previewDepth + 1
```

3.3. Operationalisation Plan Selection Function

When creating our plan selection function, establishing a simple and efficient method of operationalisation comparison was the main focus. However, we also had to adhere to the dynamic environment our agent was developed to thrive in. This means that, as well as handling the comparison between operationalisations, we also needed to refresh their values in the scenario that a change occurred. We achieved this by separating our function into two main areas. The first handles the reinitialisation of an operationalisation. This consists of creating a summation of the operationalisations contributions (multiplied by the recipient softgoals weighting) as well as the preferences of its parent operationalisations. Its preference is then set to this summation value. If any changes occurred prior to the execution of this operationalisation, then they will be captured within this new value. The second area is the actual comparison of the value itself. This process gets the operationalisations preference value, multiplies it by the probability and then multiplies it again by its preview value. This results in the comparison value which takes both the probability and quality of the operationalisations children into account. This value is then compared with the same computation from other operationalisations and the one with the highest value is selected. The pseudo code for this plan selection function can be seen in Algorithm 2.

Algorithm 2 OptimisedPlanSelection(OP

```
Input: OP: Operationalisation being considered
Output: preference: Operationalisation score, influenced by its child operationalisations
1: $preference = 0.0$
2: for each $c \in OP.contributions$ do
3:     preference = preference + (c.softgoalWeight $\times$ c.value)
4: end for
5: for each $op \in OP.parents$ do
6:     preference = preference + op.score
7: end for
8: return (preference $\times$ probability) $\times$ previewValue
```

3.4. ENFR Illustration

Considering our approach, the current way of representing operationalisations within the ENFR diagram no longer provides enough detail to make the graph readable and understandable. In sight of this, a new representation was created using a rectangular form. This form displays the operationalisation's name along with the probability, preference and preview values underneath respectively. An example of this new form can be seen in Figure 1.

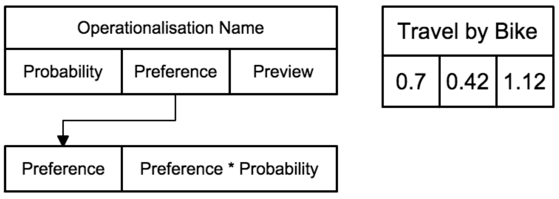

Figure 1. ENFR Node Example

4. Optimisation Model

This section develops an optimisation model for ENFR framework models based on the optimisation approach currently used in this methodology. It explores each component of our approach mathematically, detailing how they are used in collaboration to select the optimal operationalisation.

4.1. Model Calculations

For a given ENFR graph G, the associated variables are calculated as follows:
Base Preference: The preference of an operationalisation without the preview value being taken into consideration. The calculation is based on the operationalisation's contributions to softgoals (respective to their weighting) as well as any parent operationalisations it may have. Equation 1 calculates the base preference with Equation 1a calculating the original preference value and Equation 1b applying the probability to it.

$$basePref = \sum_{SG_i \in SG} (contr_i \cdot W_i) + \sum_{pOP_j \in OP} basePref_j \tag{1a}$$

$$basePref = basePref \cdot probability, \quad 0 \leq probability \geq 1 \tag{1b}$$

where SG is a set of leaf-softgoals, W is the softgoals weighting and pOP is a set of parent operationalisations.

Preview Value: The preview value is calculated using two variations to account for the inclusive and exclusive relationships between operationalisations existing within G. The propagation elements of these variations are shown below with a generalised preview calculation shown after in Equation 4.
AND: Equation 2 calculates the AND propagation elements of the preview value. Equation 2a cycles through the operationalisation's children to calculate a cumulative average amongst them. Equation 2b does the same but in relation to the children's depth, determining how many child operationalisations exist through out the relationship.

$$preview = \sum_{cOP_i \in OP_j} (comparisonPref_i \cdot previewDepth_i) \tag{2a}$$

$$previewDepth = (\sum_{cOP_i \in OP_j} previewDepth_i) + 1 \tag{2b}$$

where cOP is a set of child operationalisations belonging to a parent operationalisation.
OR: Equation 3 calculates the OR propagation elements of the preview value. Equation 3a cycles through the operationalisations children to determine the maximum comparison value amongst them. Equation 3b calculates the preview depth value based on the operationalisation selected previously.

$$preview = max(comparisonPref_i \cdot previewDepth_i), \quad \forall OP_i \in cOP \tag{3a}$$

$$previewDepth = previewDepth_i + 1 \tag{3b}$$

Generalised Preview: Equation 4 calculates the final preview value and preview depth value for an operationalisation j based on the above propagation elements. Equation 4a calculates the preview value for the operationalisation. Equation 4b calculates the preview depth for the operationalisation.

$$preview_i = (\frac{preview + basePref_i}{previewDeph_i}) \ / \ basePref_j \tag{4a}$$

$$previewDepth_i = previewDepth \tag{4b}$$

Comparison Value: The Comparison value is calculated based on the preview value and the base preference of the operationalisation, as seen in Equation 5.

$$comparisonPref = basePref \cdot preview \tag{5}$$

Attainment: The attainment score represents a system where not all softgoals can be completely satisfied. Given perfect satisfiability is unachievable, the highest possible satisfiability must be used instead, which is achieved through contributing operationalisation i, as seen in Equation 6.

$$comparisonPref_i = Max(Min(comparisonPref_i, 1) - 1), \quad \forall LSG_j \in Selection \tag{6}$$

$$A_{actual} = \sum_i (LSGw_{(j)} \cdot (contr_{(ij)} \cdot comparisonPref_i))$$

where $contr_{(ij)}$ is a contribution from operationalisation i to softgoal j and $Selection$ is the series of softgoals relevant to the current environment of the agent.
In order to achieve the full contribution, $comparisonPref_i$ must be equal or larger than one. Any value less and the full contribution will not be achieved.

Through this model, the attainment can be reduced to the average of the base preferences starting from the initial operationalisation such that:

$$max(A_{actual}) \propto max(\ Avg(basePref)\ ), \quad \forall cOP \in pOP \in Selection$$

Therefore, in order to achieve the maximum attainment for the system, the operationalisation and its children which have the highest base preferences must be chosen.

5. Evaluation

Now that we have detailed our optimisation model and how the agent can use it to achieve optimality, we evaluate our optimised approach through empirical experimentation. Our approach, based on the operationalisations score relative to its children, should always choose the best path of plans regardless of the underlying ENFR models configuration. To test this declaration, we perform static, random, preferential and our optimised plan selection approach on a set example. The settings of this experiment are shown in section 5.1 with the results and prompt discussion shown in sections 5.2 and 5.3 respectively.

5.1. Experiment Settings

Our experiment consists of a simulation that compares the satisfaction achieved in the system when a specific plan is selected using each of the methods, beginning with an untouched ENFR model configuration. This configuration will then be randomly changed through out the agents execution in order to emulate a dynamic environment. The evaluation will consist of four distinct approaches. The first is a static approach which comes in two variations, one which always selects the first operationalisation and another which always selects the last. The second approach is completely random, selecting a random operationalisation from the available selection, regardless of scoring. The third approach is the preferential approach which selects an operationalisation based on its preference alone, independent to its children. The final approach is our optimised approach, which selects an operationalisation based on its preference and the preferences of its children.

The simulation scenario is very similar to the banking system seen within the ENFR literature [1, 2], except it has been drastically extended to allow for a more stressful evaluation. Throughout this simulation, the agent must constantly make decisions upon which operationalisations it should execute to ensure the success of the system in relation to a series of softgoals. These softgoals are as follows:

Space: Maximising the space maintained by the system so that the system can be sustained for longer durations without needing to be upgraded.

Response Time: Minimising the response time of the system so that accessing information within the bank and executing various queries can be done in less time.

Data Security: Maximising data security means that the banks information, including credit cards numbers, fund access and transaction history is properly protected.

Accuracy: Maximising accuracy results in user access to the system being properly validated so that their transactions and requests only effect the required individuals.

Confidentiality: Maximising confidentiality means maintaining the access and integrity of the users information.

Memory Retainment: This is in relation to how information from the bank system is presented to its users, and how they can both remember and understand it.

Availability: This refers to how long the bank system is available for user access, which should ideally be constantly through out each year.

User-Friendly: Since consumers of varying intelligence will use the system, this refers to how easy it is for them to navigate through it and access all the functionality that they require.

Using the model defined above, each operationalisation plan has different contributions (if any) to a subsection of these softgoals. Due to the sheer size of the model, these contributions will not be explicitly stated. Our experiment consists of the following steps after the ENFR diagram generation:

1. Select an appropriate plan pathway using all prior mentioned approaches.

2. Evaluate the attainment of the selection from the system using the optimisation approach detailed in Section 4.1, which measures optimality based on the comparison value of the selected operationalisation.
3. Modify the ENFR model configuration by replacing an operationalisations contribution to a softgoal, an operationalisations probability or a softgoals weighting with a randomly generated value.

Evaluation of each approach will be based on the same triggered goals and dynamic changes in order to ensure a fair experiment.

5.2. Results and Analysis

In our experiment, we ran 1000 iterations of the steps described above. This number was chosen arbitrarily to provide an indicative, traceable measure of the performance of our approach, with the results being entirely independent to it. Each iteration took a fixed 0.1 seconds to run due to threading considerations, with the performance being well under this with the limitations removed.

For each iteration we noted the comparison value achieved by the selected operationalisation, since this value denotes the operationalisations overall benefit to the system. This information was summarised for each approach and presented in a more distinguishable form. This includes the max, min, mean, median, standard deviation and variation scores. The raw scores are shown in Table 2, with a supporting visual box-plot in Figure 2.

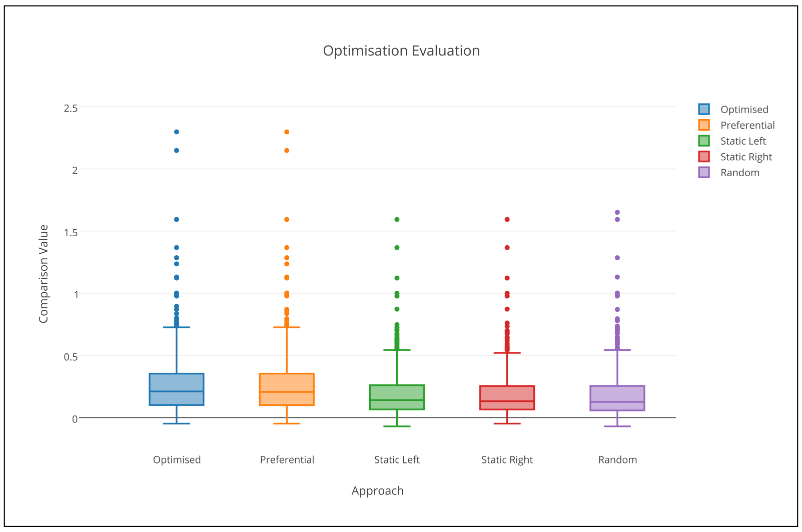

Figure 2. Evaluation Box-plot (n = 1000)

Approach	**Min**	**Max**	**Mean**	**Median**	**SD**	**V**
Optimised	**-0.04762**	**2.298**	**0.26157**	**0.21108**	0.23284	0.05421
Preferential	**-0.04762**	**2.298**	0.25664	0.20717	0.23234	0.05398
Static Left	-0.07004	1.59383	0.19182	0.14155	0.18254	0.03332
Static Right	**-0.04762**	1.59383	0.18588	0.13250	**0.17711**	**0.03137**
Random	-0.07004	1.65124	0.18657	0.12743	0.19183	0.03680

Table 2. Evaluation Results (n = 1000)

As can be seen from this information, on average, our optimised approach had the highest attainment rate, while the remaining approaches had less than satisfactory results, albeit similar. This means that, even with constant, dynamic changes being applied to the agent, our approach was still able to reliably select the best plan to achieve the maximum attainment. We will now examine the results achieved by exploring each notable measure sequentially to reinforce our approaches optimality.

The maximum and minimum values across the experiment were -0.07004 and 2.298 respectively. Our optimised approach achieved the maximum value and also achieved the highest minimum value of -0.04762. This means that, in both the best-case and worst-case scenarios, our optimised approach performed optimally, especially in comparison to the other approaches.

The highest mean value achieved across the experiment was 0.26157, which was achieved uniquely by our optimised approach and was 29% higher than the smallest average. This data indicates that, during the experiment, the operationalisations chosen by our optimised approach achieved higher attainment in comparison to all other approaches. This includes the preferential approach, which achieved a mean value of 0.25664. This demonstrates that our optimised approach surpasses a similar approach which bases itself only on the current operationalisations, irrespective of their children. Although this increase is only 2%, given larger models and a higher branching factor, this gap could increase exponentially.

The smallest standard deviation achieved across the experiment was 0.17711 from a static approach. Our optimised approach in comparison achieved a standard deviation of 0.23284, which is 24% higher. This entails that our approach is not as consistent with its values in comparison to the other approaches. And whilst this is empirically correct, within our agents dynamic environment, this inconsistency is something that is favoured. It means that our agent selected values between cycles that greatly differed from each other. Since each cycle has a different set of available operationalisations to choose from, and a new modification to the underlying ENFR model, this inconsistency is a trait of optimality. To achieve consistency would be to select the worst value in every scenario since value degradation is more consistent that improvement. Therefore, having the largest standard deviation from the experiment is supportive of the optimality demonstrated by our optimised approach.

5.3. Discussion

As discussed above, in all iterations, our optimised approach selected the optimal path. Compared to the other approaches, this means that our approach achieved the highest level of attainment consistently, given the dynamic context it participated in.

As well as optimality, our experiment also gave us a platform to compare the technical performance of the approaches. Our optimised approach had a time complexity of $\mathcal{O}(s)$, where s refers to the number of plans under consideration. This is slightly larger than the time complexity of the remaining approaches, which have a minimum time complexity of $\mathcal{O}(1)$. Given the way our agent operates however, this difference is negligent. This is because this difference is dependent entirely on the underlying ENFR model where the number of plans originates from. Given that only a subset of these plans can be selected initially (contributing plans) and a further subset of these plans exist for selection (applicable plans), even large ENFR models will have little effect in terms of time complexity. This is evident through the results, which had indistinguishable timing for each approach, aligning with the processing constraints commonly required by agent technology.

Our experiment also allowed us to identify a limitation of our approach which is with regards to the way the previews are expressed, which is as a percentage. By expressing the preview in this manner, some ENFR model changes do not require propagation to maintain preview accuracy. This is because the ratio between the new preference and its children remains static. However, when children of an operationalisation contribute to a softgoal, this addition can cause the ratio to become inaccurate. Although this is rectified naturally through agent computation, the cycles required to do so may possibly inhibit optimality by a very small amount for a brief duration. Rectifying this would cause the agent to perform propagation in scenarios where it is not necessarily required. Due to the additional computation this would require, it is currently not considered, but may be in future iterations.

6. Related Work

In this section, we discuss work related to our approach, starting with an approach similar to ours in terms of core mechanics, and finishing with a methodology relating to our process-orientated philosophy.

Softgoal-based plan selection is a softgoal-based approach to agent plan selection [8] implemented through BDI4JADE [7]. Softgoal-based plan selection uses a series of softgoals and plans, with contributions shared between them. The agent is evolved from a declared meta-model, utilising these components through a model-driven approach. There are two key differences between this and our approach. The first difference is that they do not consider their child plans when selecting a plan. Their selection is based only on what their agent can currently see and access. As we have proven, this leads to plan selection that is suboptimal, resulting in their approach selecting plans which may not necessarily be the best choice. The second key difference is the intended environment of the agent. Our approach was built around the idea of performing in a dynamic environment where percepts from said environment will constantly change the agents state. Their approach does not consider such an environment, they assume their environment is static so that their belief state will not change through the agents lifetime. Given this static set of base knowledge, their agent can be considered an abstraction of a reflex agent. These agents cannot operate in an environment unless it is static and discrete. This limits their agents drastically in comparison to ours, which is a derivative of a utility agent, such that it can cope within dynamic environments by modifying its belief state to suit any changes [12].

Our approach to designing agents is a Process-orientated one, not unlike agent methodologies such as Prometheus [9]. Prometheus is a standard planning-to-implementation approach to designing agents. Prometheus ventures into detail about creating various entities in a structured approach. This means that the objects are created in earlier stages and are progressively abstracted less and less until they can be implemented. Our approach is much more light-weight in comparison to Prometheus, along with being more accessible and easily changed without having to adhere to harsh design constraints. This however, means that our approach is also very static in the agents that it can create. Although they can fulfil many uses, they are constrained due to their model-based foundations. Using Prometheus, one can create a wider range of agents with numerous purposes, albeit with much more overhead compared to our approach, and requiring expert knowledge.

7. Conclusion

Using models as the basis for agent development allows many of the usual intricacies involved, such as the expert knowledge requirement and complicated design methodologies, to become nearly non-existent. This means that agents can reach a much wider audience, which, considering their abilities and nature, could vastly improve many existing software applications. By further optimising the way these agents interact and progress, both technical and commercial viability can be achieved.

In this paper, we have detailed our model-driven agent development approach, demonstrating its development, usage and optimisation. Our approach consists of taking an ENFR-model, and converting it into an agents belief state through our defined meta-model. The agent can then interpret and select from the operationalisations within this model. Through our approach, this selection process considers the operationalisation itself as well as its children. This produces an optimal result with respect to the current state of the agent as well as future states (defined by the operationalisations children). Our approach is evaluated empirically using an enlarged example existing within a completely dynamic environment. Our approach was required to select the optimal operationalisation path through the possible selection set with respect to any random changes that occur. In comparison to other approaches such as static, random and preferential-based, our optimisation approach achieved favourable results, performing optimally.

Future work for this approach includes the development of a graphical tool which will allow users to draw and label an ENFR model through its interface. This can then be used to directly create the code required for the agent, ready for the user to run, access and manipulate at their discretion. Another future addition is integration with the Prometheus methodology, or the creation of an entirely independent methodology which would allow planning created specifically for these types of agents. This would make their creation much more efficient and handle building the initial ENFR diagram, which is a requirement of agents created through this approach.

References

1. A. Affleck and A. Krishna. Supporting quantitative reasoning of non-functional requirements: A process-oriented approach. In *Proceedings of the International Conference on Software and System Process*, pages 88–92, 2012.
2. A. Affleck, A. Krishna, and N. Achuthan. Optimal selection of operationalizations for non-functional requirements. In *Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling - Volume 143*, pages 69–78, 2014.
3. A. Alan, E. Costanza, J. Fischer, S. Ramchurn, T. Rodden, and N. Jennings. A field study of human-agent interaction for electricity tariff switching. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 965–972, 2014.
4. C. Burgess and A. Krishna. A process-oriented approach for the optimal satisficing of non-functional requirements. pages 293–304, 2009.
5. H. Dam and A. Ghose. Automated change impact analysis for agent systems. In *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*, pages 33–42, 2011.
6. M. Huget. Agent uml class diagrams revisited. In *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, pages 49–60. 2003.
7. I. Nunes, C. Lucena, and M. Luck. Bdi4jade: a bdi layer on top of jade. *ProMAS 2011*, pages 88–103, 2011.
8. I. Nunes and M. Luck. Softgoal-based plan selection in model-driven bdi agents. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 749–756, 2014.
9. L. Padgham and M. Winikoff. Prometheus: A methodology for developing intelligent agents. In *Agent-oriented software engineering III*, pages 174–185. 2003.
10. A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A bdi reasoning engine. In *Multi-Agent Programming*, pages 149–174. 2005.
11. A. Rao and M. Georgeff. Bdi agents: From theory to practice. In *In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319, 1995.
12. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* 2003.
13. M. Winikoff. Implementing commitment-based interactions. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 128, 2007.