Towards an Ethical Machine:
One Test at a Time

by

Thomas Steven Highstead

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfilment of the requirements for the degree of

Master

in

Cognitive Science

Carleton University
Ottawa, Ontario

Abstract

Machines performing as artificial agents, such as autonomous vehicles, are becoming more frequent in society and are interacting directly with human beings. Because social interaction is grounded in ethical norms, artificial agents also need to behave in accordance with the ethical norms of the society in which they operate. To be accepted by society, an artificial agent must incorporate moral judgment in the performance of its tasks. Artificial moral agents would, therefore, be seen as safe and courteous in their daily intercourse with people. To accomplish this, requires a methodology that can communicate ethical values from an ethical domain of discourse to the scientific domain of engineering design and development. I present an innovative approach, which incorporates the concept of an oracle used to interface an ethical evaluation process to a Test-Driven Development methodology employed in the design and develop an ethical machine. The oracle is the repository of moral values received from the ethical evaluation process that, in turn, become specifications for developing a machine's moral capacity to assess its actions. By morally ameliorating an artificial agent's tasking, the machines actions assume an ethical quality. The result is that an artificial agent's actions are deemed to be morally acceptable, and therefore, it becomes an ethical machine.

Acknowledgement

Without the support and guidance of Dr. Babak Esfandiari, I don't think this thesis would have been completed. Babak introduced me to Test-Driven Development as well as software development and elementary discrete mathematics, which are fundamental skills that I needed in developing my thesis.

Dr. Rob West provided the guidance and much of the necessary information for preparing a thesis in the Cognitive Science environment. This advice has been crucial to the form and content of the final product.

I am grateful to Dr. Jennifer Schellinck for the tremendous help in editing and advice in preparing my thesis.

I am also grateful to Dr. Patrick Boily who took the time to read my thesis, and who agreed to be a member of my thesis defense committee.

I am especially grateful to Dr. Jim Davies, who acted as chair for my thesis defense.

Table of Contents

Table of Figures

Test-Driven Development

An agent is an entity that possesses the ability to act independently of any intervention by another entity. Since an agent is an autonomous entity, it can decide how it will behave in a given environment. An artificial agent is also capable of acting within an environment where their "activities do not require constant human guidance or intervention" (Shoham, 1993, p. 53). However, when an agent's actions can affect another agent the issue of morality is raised. This thesis addresses one approach to building an artificial moral agent.

The challenge that many of those who study artificial moral agency, (Anderson & Anderson, 2007) (Allen & Wallach, 2009) (Lin, Abney, & Bekey, 2011) is how can two very different domains of discourse be bridged.

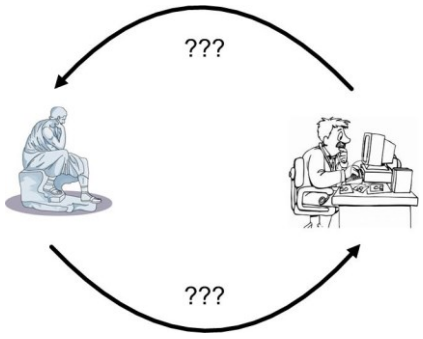

Figure 1 Two Domains of Discourse, Ethics and Science

Ethics, according to many who have a scientific paradigm, is untestable. However, the realm of ethic discourse is the source of the thinking that tries to understand how human behaviour can be morally evaluated. For a scientist, though, there is little that can be grounded in fact that can be functionally useful (Allen & Wallach, 2009). At the same time, the interests of a scientist, or an engineer, are problematic for an ethicist to appreciate, see Figure 1. What is needed is a mechanism to bridge these two realms of discourse. The suggestion I am proposing is to convert ethical premises into factual data

that is useful to a scientist. The conceptual device I am proposing is to employ an Oracle as depicted in Figure 2.

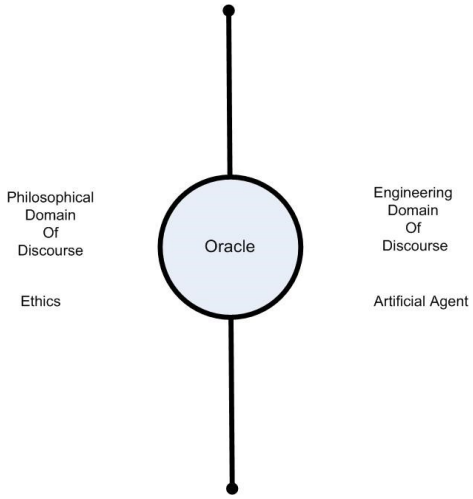

Figure 2 An Oracle as an interface between Ethics and Engineering

An oracle takes ethical concepts and presents them as testable rules or specifications that can be used in developing moral software employed to ameliorate an artificial agent's actions. The specifications made available by the oracle can be employed by methodologies used in Test-Driven Development. A discussion of the role of the oracle is presented later.

The methodology of Test-Driven Development was first formalized by Ken Beck in his book *Test-Driven Development by Example* (Beck, 2003). Subsequently there have been additional resources that have become available such as David Astels' *Test-Driven Development: A Practical Guide* (Astels, 2003). In the Test-Driven Development paradigm, software specifications for functionality and features are first formulated as a test. The test is then performed on the software. If the test fails, which it should initially since there is no software code that implements what is being tested, software is written and tested until such time the test is passed. The software code can then be refactored and

cleaned to remove any duplication and make the code more readable. Once the test is passed, the development cycle begins once more starting with the addition of a test. In the Forward to Astels book, Ron Jefferies writes, “By focusing attention on tests first, you'll be designing your program more from the viewpoint of the user. By doing the tests one at a time, you'll be creating a simple design that's focused exactly on the problem” (Astels, 2003, p. xii). This is reversed from common software paradigms, where software code is first written then tested.

Background to Test-Driven Development

Test-Driven Development follows the Popperian notion of *falsification*. Falsification is the concept that a theory or hypothesis does not have scientific validity unless it can be falsified. As Stephen Thornton explains in his Stanford Encyclopedia of Philosophy entry on Popper,

> “Popper’s theory of demarcation is based upon his perception of the logical asymmetry which holds between verification and falsification: it is logically impossible to conclusively verify a universal proposition by reference to experience (as Hume saw clearly), but a single counter-instance conclusively falsifies the corresponding universal law” (Thornton, 2017, p. 9).

Thornton is referring to David Hume’s skepticism resulting from inductive reasoning (Hume, 1739/1978). A theory in science is often developed from observations. The problem of demarcation states that, if the scientific results are not questioned, that is, a test developed that can demonstrate that the research result is wrong, then according to Popper it is considered pseudoscience (Thornton, 2017). Pseudoscience is often accused

of not being science because it constantly verifies results rather than trying to demonstrate that they can be falsified. In pseudoscience, verification of theories such as Freud's theories of psychoanalysis, or astrology, can often be erroneous in their conclusions since they have not been subjected to tests that could falsify them. So, for something to be considered a scientific proposition, that proposition has to have a test to demonstrate that it can be false.

A scientific theory or hypothesis can never be proven by repeated tests that verify the results. David Hume realized this when he questioned inductive results from observation (Hume, 1739/1978). One famous example that elucidates the failure of verification is the proposition that "all swans have white feathers". Using a numerical inductive research methodology, testing the hypothesis that all swans have white feathers, so long as the testing occurred in Europe. However, while this was the case in Europe, when colonists arrived in Australia, they soon discovered swans can also have black feathers.

The reason for this is that there is an inherent fallacy in inductive reasoning referred to as "Confirming the Consequence". A famous example is Bertrand Russel's story, "Domestic animals expect food when they see the person who usually feeds them. We know that all these rather crude expectations of uniformity are liable to be misleading. The man who has fed the chicken every day throughout its life at last wrings its neck instead, showing that more refined views as to the uniformity of nature would have been useful to the chicken" (Russell, The Problems of Philosophy, 1912/2010, p. 44).

As Hume, and subsequently Popper proposed, an inductive result can be proven wrong, because the result can simply be otherwise. Popper reasoned that the only way to avoid the results of an investigation from being considered pseudoscience would be to, at the same time, demonstrate what would prove the theory wrong (Thornton, 2017).

While Test-Driven Development is normally associated with software development, since it embodies the Popperian concept of falsification, its application can fall into other domains of discourse. Any theory, hypothesis, concept or notion can be questioned. An author of a hypothesis or theory can test their concept against the question. If the concept holds up to the questioning (testing), then nothing further is required. However, if the concept fails the test, i.e. is falsified, then work is required to improve the concept until such time it can pass the test. The revised concept that passes testing would quite likely be messy, which would then necessitate that the revised concept be cleaned up, or in Test-Driven Development parlance, "refactored." For this reason, Test-Driven Development has been chosen as the methodology, which I believe is most effective in dealing with an ethics discourse involving artificial agency.

Ethics and Test-Driven Development

Ethics and morality (ethics is of Greek origin; morality of Latin origin) are two words which mean essentially the same thing, *practical reasoning* about what one should do in a given situation. While the usage of the terms have been refined over time (ethics is the study of right action, morality is an expression of right action, one's duty) for my purposes ethics deals with norms, while morality deals with the execution of a norm. The subject of ethical discourse is as diverse as is humanity. Later I shall attempt to limit the

range of ethical concepts, but for now, the scope of possibilities, and discussion lends itself to applying the concepts of Test-Driven Development.

Inherent in Test-Driven Development is the notion that there is a standard of performance that a test ought to pass. If what is being tested is a device, or an agent, which is purported to act in a morally "good" manner, then there must be a standard of "goodness." The test has two parts, what is considered "good", and what the entity actually does. The perennial debate, of course, is what standard of goodness one should use.

The standard of goodness can also be tested, or so one would think, but there is a severe problem with equating a natural phenomenon with what is accepted as good. There is an inherent "naturalist fallacy" (Moore, 1903/2012) whereby one cannot logically infer a "good" from physical/natural properties. So, what is one supposed to do if one cannot "prove" that a good is such? Using Test-Driven Development, one could build a body of data that can be argued as being good. The standards for testing could be such concepts as autonomy, beneficence, non-maleficence, and/or justice. However, these concepts have to be agreed upon before any testing can proceed.

Assuming that there is an ethic that can be generally accepted, these norms can then be used in practical reasoning: reasoning about what would be the "right action" as Aristotle would opine (Aristotle, 1984). The right action would draw upon the body of ethics that would determine the best moral behaviour. Test-Driven Development practices could be applied to making the determination of the right action in a given environment. The test standard, the Ideal Observer (discussed in a later section) to use meta-ethical

terminology, provides the expected result of a moral deliberation, and this result is compared against the action taken by an agent.

For example, suppose a person is travelling along a highway, and they come across a person injured by the side of the road. What should the traveler do? If they stop, it could be a trap, and they could be robbed or worse. Thus, they could take the attitude that they should not get involved, and continue their trip. This they could also stop, take that chance it is not a trap, and lend assistance to the injured person. All of these are ethical choices, and what one does depends on which "standard of goodness" one follows. If the standard of goodness states that one should always stop and lend assistance, then to pass a Test-Driven judgment, one would not continue their travels, but stop and lend assistance to the injured party. In this example, Test-Driven Development methodologies, provide a standard of performance, and compares that standard against the action taken by the agent. If the agent acts in concert with the standard of performance, then the agent has passed the test. (The Good Samaritan refers.)

In developing artificial moral agents, Test-Driven Development provides the framework to build an agent that embodies ethical norms that can be employed in a social environment, provided, of course, the ethical norms are consistent with the society's norms. An artificial moral agent could also be designed to hold global ethical criteria (as opposed to specific deontological duties) such as "do no harm" and evaluate given social environments to determine their actions. A Test-Driven Development framework that tests for specific environmental situations could refine the artificial agent's global ethical criteria to match the behaviours specified in the test.

As ethical criteria are added, the set of norms could be refactored to remove conflicts and duplication. The importance of refactoring is to make the body of knowledge available to an artificial agent cleaner and easier to understand. In normative terms, there could be a rationalization of virtue ethics, deontological ethics and utilitarian ethics into a comprehensive and flexible ethical model for artificial moral agents.

Software Development and Test-Driven Development

The classic software development paradigm follows a Software Development Life Cycle (SDLC) often referred to as the "waterfall" method for software development. It follows the classic engineering paradigm of collecting all pertinent requirements, analyzing the requirements, and designing the project based on requirements, in order to produce specifications. The projects are then developed based on the final sets of drawings and specifications. In this method of software development, the software testing occurs at the end of this process. While this system works for physical engineering projects such as buildings and bridges, it is less effective in software development (Beck, 2003). The reason is that software development has quite volatile requirements with specifications changing frequently throughout the design and build program. When the "waterfall" method is employed, for a change in requirements to be implemented, the full design needs to be reviewed and modified in order to incorporate the change. This results in development delays, which retards the final release of the software under development. While the classic systems development approach is still used in engineering, in software engineering, this approach has been replaced by a more agile approach to software design paradigm.

The concepts I employ with respect to test-driven development are best expressed by Kent Beck (2003) in these five steps:

1. Quickly add a test
2. Run all tests and see the new one fail
3. Make a little change
4. Run all tests and see them all succeed
5. Refactor to remove duplication

The first step is to write a test before any code has been written. Test Driven Development follows this philosophy since it is constantly testing code to determine if it can fail. That is why the first effort is to identify and write a test for the software specification that the software code has to pass. This demonstrates that if the software code passes the test, then it is software code that is a sound example of the specification.

Initially, the test will fail, since there is no code for the test to pass. The next step is to write some code, test the code, write some more code and repeat until the test can be passed. The cycle of test and software code writing continues until code becomes what some refer to as “smelly” at which point the software is refactored. The refactored code then is run through all the tests to make sure that it doesn’t fail. At this point, the cycle of test and code writing is repeated, adding functionality to the code base.

The advantage of this approach is that if specifications change, then developers can respond quickly to code revisions. In the “waterfall” paradigm, if specifications are changed, it necessitates full review of all specification to analyze and determine the changes have on the design. This allows Test-Driven Development to be more responsive to client requirements.

All tests are retained, and when a new feature passes its test, and particularly after refactoring, all tests are run on the software code. This ensures that any new features, or

refactoring, do not break previously developed code. This is referred to as regression testing.

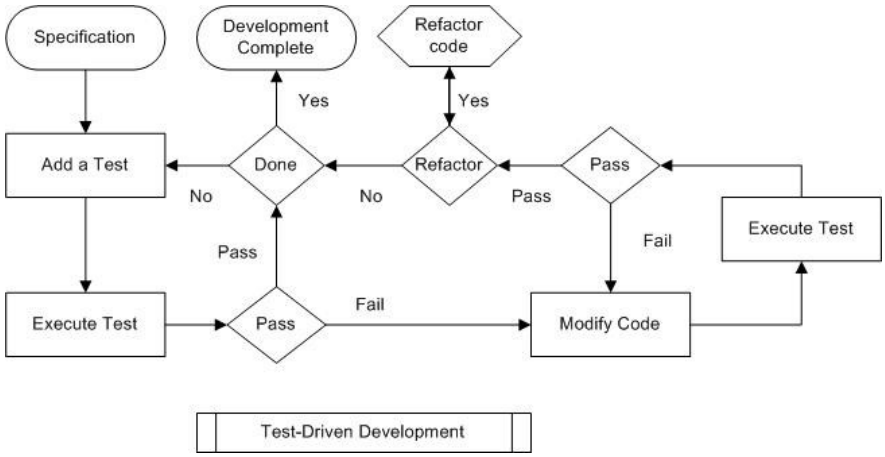

Figure 3 Test Driven Development Life-cycle

Figure 3 is a schematic of my interpretation of Kent Beck's approach to software development. The "specification" identifies what the software is required to return. This specification is formulated as a test. The first execution of the test will result in a failure, or a "Red Bar" indication of a test failure since there is, initially, no software code (the Red Bar is the indication returned by the Java Test Suite I employed in developing the Ethics Class, the Junit Test suite).

If the code fails, then the code is modified and tested again. If it passes, a "Green Bar", then the development sequence passes a check to see if the code needs to be cleaned up, i.e "Refactored". Next is a check to determine if the development is complete. If these steps indicate further work is required, then a new test is written to advance the development of the software. This has been expressed in a mantra, "Red, Green, Refactor", which summarizes the development cycle. Red refers to the Red Bar failure of a test, while Green refers to the Green Bar when a test passes. Refactoring refers to cleaning software code of redundancy, and making the software more readable.

Testing in Test-Driven Development

As it is in science, testing is fundamental to test-driven development. Testing, especially self-checking tests (Fowler, 1999), reduce time in finding bugs in software. Every time a function or feature is added, a test is also added to the software. Using the test-driven development methodology, tests are developed prior to software development. As the software is developed, so is the test suite for the software. As functionality is added to pass the test, all previous tests (regression testing) are run to ensure the software meets specifications as well as the new functionality just added (Astels, 2003) (Beck, 2003).

When legacy software is to be modified, if there is no test suite available, then the first step is to write a suite of tests. If code is abstruse, then it should be refactored before any work on the software occurs (Fowler, 1999). Regression testing will reveal any errors resulting from initial refactoring. These tests, which can use *mock objects* (Beck, 2003) (Astels, 2003) (instead of *real* objects), should produce the same results as the actual results produced by the legacy software (Fowler, 1999). By taking the time to build a test suite, when the software is refactored, running the tests will highlight errors resulting from a refactor that has been implemented. Adding features or functions to the software then follows the test-driven development paradigm by writing tests first, adding the software to pass the test, then refactoring the new software.

A test can be understood as writing the software specification as a question. For example, if the software is to add two numbers, then the resulting sum must be known prior to any software being written. Given the number 5 and the number 7, the software, when written correctly, returns the value of 12 (*pace* Kant).

The test for two numbers would look something like the following Java "assertTrue (expectedValue, returnValue)" test statement. The "expectedValue" is what the test requires the software code to return. The "returnValue" is what the software actually returns. In this example, where the sum of 5 and 7 are required; the "expectedValue" is 12 and for the test to be successful the "returnValue" must also be 12.

If we accept the definition for an "oracle" to be "person or thing regarded as a source of wisdom" (OED), then the value of the "expectedValue" is the "wisdom" that the software code will return. It is the standard, the "the source of wisdom," to which the software must adhere in order to pass the test. So, the oracle provides the expected value for the test.

Refactoring in Test-Driven Development

In the test-driven development methodology, refactoring is defined as "a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior" (Fowler, 1999, p. 53). A refactor method is one of a number of techniques developed by Fowler, Beck and others that make small changes to software.

Refactoring does not intentionally enhance performance or add functionality to software. Its main purpose is to rewrite software so that it is more easily understood. "Refactoring helps you to make your code more readable" (Fowler, 1999, p. 56). Good software code should communicate its purpose and function. "Any fool can write code that a computer can understand. Good programmers write code that humans can understand" (Fowler, 1999, p. 15). Good software should be readily understandable. One of the benefits of refactoring early in software development is that certain aspects of the

program design become clearer. For example, by removing duplication, there is less chance of failure when software code is modified. With code duplication, all variants of the duplication need to be picked up for the modification to function properly. Removing duplication allows for easier software modification.

Refactoring is distinct from writing new software functionality. Kent Beck is attributed with the two hats metaphor to practicing test-driven development (Fowler, 1999). The first hat is when a programmer is writing software to pass a test by adding functionality to the code base. This is not refactoring. The programmer then wears the second hat once the test has been passed, and the software code is refactored and cleaned up. (Fowler, 1999) (Astels, 2003). Each hat is independent of the other, since their purposes are distinct from each other.

Refactoring changes the emphasis on the design of a system by modifying the design as the system develops. What is desired is not a full design but a "reasonable solution" (Fowler, 1999). As the problem becomes better understood, and the system development matures, refactoring allows for a better solution to emerge by changing and improving the design. If refactoring was not employed, then more emphasis is placed on the design which becomes more difficult to change as the system matures.

Refactoring simplifies design by shifting emphasis away from comprehensive design, and designing flexibility to account for possible variations in requirements. While flexibility and other design considerations are kept in mind, allowing for refactoring as the system develops also allows for a simpler designs.

Conclusion

While Test-Driven Development is an effective way to create code, for the study of morality, it is the philosophy behind the method that is attractive. If the resulting software code is considered a theory, then the questioning (testing) of the theory is the core determination of the validity of the theory. As discussed above, this approach is applicable to numerous problem solving situations, and provides confidence that once the question is answered correctly, then the underlying response to the question is correct.

.

Modelling an Ethical Dilemma

In this chapter, I present the ethical dilemma that I use to demonstrate Test-Driven Development of an artificial agent, and I present a model for the dilemma. Since I am testing the agent's moral judgment, I employ a classic moral dilemma known as the Trolley Problem (Foot, 1967) (Thompson, 2009) which an artificial agent needs to solve. I first introduce the Trolley Problem, and explain the ethical dilemma it presents. Next I present a model that an artificial agent can use in attempting to solve the dilemma.

Ethical Dilemma – Trolley Problem

Ethical dilemmas take many forms, but usually all are grounded in making a choice between a morally acceptable act, and one that is less morally acceptable. The Trolley Problem is a classic example of a moral dilemma, and provides an elementary challenge to moral reasoning. It is challenging, because of the conflict it causes within people when choosing the best course of action to take in a dire circumstance. The Trolley Problem expresses a clear set of choices that conflict with our intuition regarding what is a "good" act and a "bad" act. The choice in the Trolley Problem applies to many scenarios in life where people are wedged between a "rock and a hard place," so, choosing the Trolley Problem as a vehicle to develop concepts for artificial agent ethics allows for a basic challenge to moral judgment.

The trolley problem was first formulated by Philippa Foot as the "runaway tram" (Foot, 1967), and was also discussed by Judith Jarvis Thompson's in her paper "The Trolley Problem" (Thompson, 2009). Thompson presents the two most common scenarios as follows:

1. A trolley has lost its brakes and cannot stop. There are five people on the track out of sight of the trolley who cannot get off the track in time before the trolley hits and kills them all. The track has a spur line onto which the trolley can be switched, but there is also one person on the track and out of sight of the trolley. If the trolley is switched to the spur line, five people are saved but the one person on the spur line is killed. You have been given control of the lever that can switch the trolley to the spur line. The dilemma is: do nothing and allow five people to die, or switch the trolley to the spur line where one person will be killed.
2. A trolley has lost its brakes and cannot stop. There are five people on the track out of sight of the trolley who cannot get off the track in time before the trolley hits and kills them all. You are standing on a bridge over the track with a Fat Man and you realize the trolley is out of control. If you push the Fat Man onto the tracks, you know he is large enough to stop the trolley. If you do this, the Fat Man will be killed, but five people will be saved.

Both scenarios have the same result. One person is killed to save five people. But why is killing the Fat Man considered more repugnant than activating a lever and killing someone on the spur line? Both results have the same utility. Philippa Foot explains that the difference is due to the Principle of Double Effect first presented by the Roman Catholic theologian, Thomas Aquinas (1225-1274). If we act to kill someone intentionally, as would be the case with the Fat Man, this would be morally wrong. However, if we intentionally act to save five people with a foreseeable but unintentional consequence of killing one person, this is more morally acceptable (Foot, 1967). There is another distinction between the two cases; should one act to kill someone, or should one

do nothing and let people die. Deciding to act, and deciding to do nothing are both ethical decisions (Lin, 2013). Deciding which to do is another dimension to the trolley problem.

Ethical Dilemma Model for the Trolley Problem

To demonstrate the implementation of an agent that responds to an ethical dilemma, I have devised the following model which I implement using the Java programming language, described later. For now, I explain the model in detail. Rather than using the term trolley, I revert to the use of the term "tram" as originally stipulated by Philippa Foot. The mathematical constructs I am using to model the trolley dilemma were developed from formulae found in Michael Wooldridge's *An Introduction to MultiAgent Systems*, (Wooldridge, 2009).

The Tram System Environment Model. The model of the tram system environment depicted in Figure 4 diagrams the classic Trolley Problem. The tram system environment includes trams stops referred to as locations (l_n), people who may or may not be collocated at tram locations represented by the tuple, (p_n,l_n), a track switch to route a tram {a,b}, and two agents, a Tram Agent (AgT) and a cognitive agent that I have named "Sophy" (AgS).

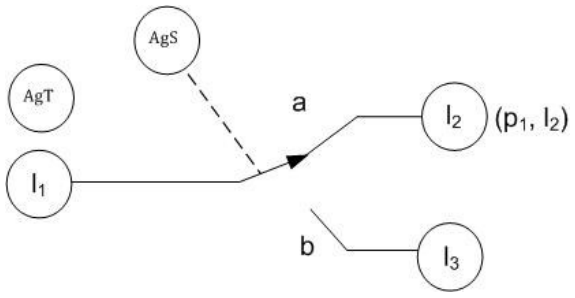

Figure 4 Tram System Environment Model

Referring to Figure 4, the Tram System environment state has elements drawn from the following sets:

Locations L ≔ {l_1, l_2, l_3}. Identifies locations in the model.

Track Switch S ≔ {a, b} Identifies the Track Switch state.

People P ≔ {p_0, p_1 … p_n} identifies people who are located at locations in the Trams System. There can be any number of people at these locations, including nobody.

PxL ≔ identifies the people at locations

A reactive agent is defined in Wooldridge as, A_g: E → A_c, where the agent's domain is the set of relevant environmental states E = {e_0, e_1, e_2, … },which are mapped to the set of actions A_c = {α_0, α_1, α_2 ... } available to the agent. This means that given one of the environmental states determines what action the agent will execute. There are two agents in the depicted tram environment model:

AgT ≔ the Tram Agent

AgS ≔ the Cognitive Agent "Sophy"

An Agent (AgS) has control of a Track Switch which can route the Tram Agent (AgT) from one track to another. In the case of the model represented by Figure 4, there are three tram stations denoted as location l_1, l_2, l_3, the Track Switch position is a, people at location l_2. Collocation of people and locations occur at any location within the system except the start location l_1. If the Tram is not routed through Track Switch set to b, then the Tram is routed through Track Switch set to a, and runs over and kills the people at location l_2. If the Agent sets the Track Switch position to b, then the people at location l_2 will be saved, and the tram terminates at location l_3. While this is the basic scenario, it can become more complicated.

Environmental States. Figure 4 represents an initial environment state e_0 where the Track Switch position is set to a, and there is one person, p_0, at location l_2. Also

represented, for completeness, are all locations L = {l_1, l_2, l_3}, the Track Switch positions S = {a,b}, and the two agents: the Tram agent (AgT) and the Sophy agent (AgS).

The set of all environmental states in this model is represented by

$E \coloneqq S \times \mathcal{P}(PxL)$

A specific environmental state, e_0, as depicted in Figure 4, is expressed by the tuple, $e_0 = (a, \{(p_0, l_2)\})$, where the Track Switch position 'a' is a member of set S, and a person, p_0, collocated at l_2, identified by the pair (p_0, l_2), is a member of the set $\mathcal{P}(PxL)$. If there were three people at location l_2 then this would be represented by the tuple, $e_0 = (a, \{(p_0, l_2), (p_1, l_2), (p_2, l_2)\})$.

The Tram System Agents. An artificial agent is a software entity, which is situated, and operates within an environment to accomplish and further its goals by performing actions on an environment (Jennings, Sycara, & Wooldridge, 1998). Aside from the characteristics that exist within an environment, an agent also possesses the following qualities:

An artificial agent, like any other agent, is *autonomous*. This means that an agent can go about its business without any direct human intervention in its activities. Autonomy also means that the artificial agent has control over its own internal states, as a result of the artificial agent evaluating its environment.

An artificial agent is *responsive* to its environment. In order to be responsive, the agent must be able to sense its surroundings (what Wooldridge refers to as the *see* function) such that it can evaluate and respond to its environment in an appropriate and timely fashion.

An artificial agent is *pro-active* which means that it has an opportunistically goal oriented behaviour and acts on situations by taking the initiative to further its goals.

Finally, an artificial agent is *social*. This means that it will communicate with other agents to solve problems and may assist them in achieving their goals.

An Agent function is defined in Wooldridge as, A_g: $\boldsymbol{\mathcal{R}}^E \rightarrow A_C$, where an agent is a function, which takes a set of runs (defined below) that end in an environment state in a set of possible environmental states, E, and maps them onto the set of possible actions, A_C, available to the agent.

To understand the set of runs, $\boldsymbol{\mathcal{R}}$, it is first necessary to define a run, *r*. A run is defined in Wooldridge as, r: $e_0 \overset{\alpha 0}{\rightarrow} e_1 \overset{\alpha 1}{\rightarrow} e_2 \ \ldots \overset{\alpha u}{\rightarrow} e_u$, where a run is represented as a sequence of interleaved environment states and actions occurring on the environment states.

The set of all possible runs is identified as $\boldsymbol{\mathcal{R}} \coloneqq \{r_0, r_1, r_3, \ldots.\}$

A run that ends in an environment state is defined as,

$$\boldsymbol{\mathcal{R}}^E \coloneqq \{r \in \boldsymbol{\mathcal{R}} \mid r\colon e_0 \overset{\alpha 0}{\rightarrow} e_1 \overset{\alpha 1}{\rightarrow} e_2 \ \ldots \overset{\alpha u}{\rightarrow} e_u\}$$

The agent I am employing does not take into account previous environmental states, but is a reactive agent that reacts to a specific environmental state. This why the agent function I am using can be expressed as A_g: $E \rightarrow A_c$.

The Tram Agent. The Tram agent (AgT) has only one action, to move from one station to the next. No deliberation occurs for the Tram agent. The set of actions available to the Tram agent contains one element: "go". The available actions, Ac_T, for the Tram Agent is defined as,

$$Ac_T \coloneqq \{\alpha \in Ac \mid \alpha = go\}$$

The Tram agent function is, Ag_T: E → {go}, since the only element in Ac for a Tram is "go". This means that a Tram agent acting on a set of environmental states, results in only one action, which is for the Tram to "go" from one location to the next in an environment state *e*

The Sophy Agent. The Sophy agent (AgS) is a cognitive agent, which acts on judgements with respect to the Trolley Problem in the Tram System. The agent will search the environment for people that are in danger. For example, if there are three people $\{p_0, p_1, p_2\}$ at Location l_2, $\{(p_0, l_2), (p_1, l_2), (p_2, l_2)\}$, the Sophy agent may or may not act. Referring to Figure 4, if the agent does not act, the Track Switch remains unchanged as a, then the Tram Agent's action runs over the people at Location l_2. This is represented as removing people at locations from the environment. For example if the environments begins as e_0 = (a, $\{(p_0, l_2)\}$) and the Tram agent kills the person collocated at location 2, l_2, then the new environmental state would be e_1 = (a, {}).

The Sophy agent (AgS) is a reactive agent, which means it reacts to the current state of the environment. The agent has two options,

Action A = set Track Switch to a

Action B = set Track Switch to b

The Sophy agent function is AgS: E → {a,b}, where the only two actions available to the Sophy agent is to choose to set the Track Switch to a, or to set the Track Switch to b. So, the Sophy agent has the set of actions Ac_S = {a, b}.

Note, if the Track Switch is set to a, and Sophy selects Action A, this is the equivalent to an agent action of "do nothing."

Referring to Figure 4, if the Sophy agent (AgS) chooses to act, then the Sophy agent selects Action B so that the Track Switch state to b. The Tram's action then routs the Tram to Location l_3, and since there are no people at this location, then nobody is killed. This would be represented with the same initial environment state, $e_0 = (a, \{(p_0, l_2)\})$, but the resulting new environment state is represented as, $e_0 = (b, \{(p_0, l_2)\})$.

State Transformer Function

When an agent acts on a given environment, depending on what the action is, it will determine the state of the resulting environment. The function that effects the change on the environment due to the action of an agent is denoted by the *state transformer function*. Wooldridge makes a reference to *Reasoning about Knowledge* (Fagin, Halpern, Moses, & Vardi, 1995) as the source of the concept of the state transformer function. Fagin *et al*, (1995), identify, what they refer to as the "*global state transformer* τ" as "a function mapping global states to global states" where "joint actions cause the system to change state" (Fagin, Halpern, Moses, & Vardi, 1995). Wooldridge refines this concept where "global states" become runs, $\mathcal{R}$, defined earlier. The state transformer function is the function that changes an environmental state from an initial state to a new environmental state due to the action(s) taken by agent(s) on the environment. For example, discussed later, transforming the initial environmental state from $e_0 = (a, \{(p_0, l_2)\})$ to a resulting environmental state $e_1 = (b, \{(p_0, l_2)\})$, is the result of the transformer function's effect on the environment. Wooldridge follows Fagin *et al*, (1995) by continuing to identify the function by the Greek letter τ. From this, the state transformer function, τ, is defined as the mapping of a set of runs to a set of possible environmental states; an environmental state e which results from an action α performed on the

environment. A set of runs that ends in an environmental state e is identified as $\mathcal{R}^{E}$ and a set of runs that end in an action α is represented as $\mathcal{R}^{Ac}$.

The state transformer function τ is expressed as, τ: $\mathcal{R}^{Ac} \rightarrow \wp(E)$, where the state transformer function maps from a domain that contains runs that end in actions, α, to a range containing all possible environmental states, e. In this variant, the resulting environmental state for a run is *non-deterministic*. With this in mind, the state transformer function can be expressed as, $\tau\ (e_0, \alpha) = e_1$, where an action α one environment state e_0 transforms or produces an environment state e_1.

In the Tram System Environment, the two agents acting on the same environment are represented in Figure 5:

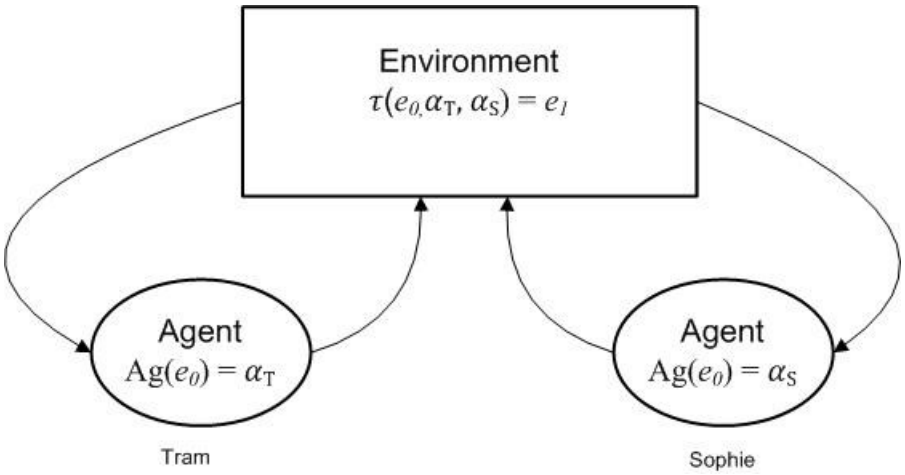

Figure 5 Agent relationship to the environment

From this, the set of runs ending in an action $\mathcal{R}^{Ac}$ is defined as,

$$\mathcal{R}^{Ac} := \{r \in \mathcal{R} \mid r: e_0 \overset{\alpha 0}{\rightarrow}\}$$

The state transformer function using two agents can be expressed as follows:

τ:E x $\mathcal{R}^{AcT}$ x $\mathcal{R}^{AcS}$ → E, where Ac_T represents the set of actions available to the Tram agent (which in this case is one: "go"), and Ac_S represents the actions available

to the Sophy agent (which are {a, b}). Since these are reactive agents, the agents act only on the current environmental state where the resulting environmental state is in the set of possible environmental states E, which can result from agentive action. (Wooldridge, 2009).

Since both Agents operate on the same environment, this can be expressed as,

$\tau\ (e_0, \alpha_T, \alpha_s) = e_1$, where $\alpha_T \in Ac_T$ represents an action from the set of actions Ac_T, that are available to the Tram, and $\alpha_s \in Ac_S$, represents an action from the set of actions Ac_S available to the cognitive agent Sophy and $e_0 \in E$ represents the initial environmental state, and $e_1 \in E$ represents the resulting environmental state.

The result of the agent's actions on the Tram System environmental state, *e*, resulting from the state transformer function is represented in Figure 6,

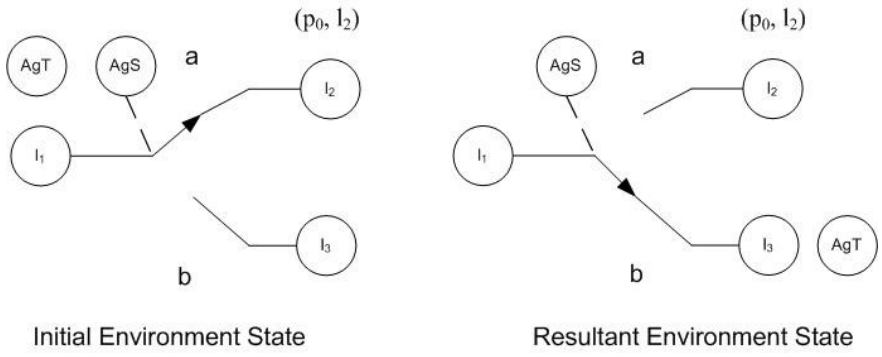

Figure 6 State transformer function operation on an environment state

Referring to Figure 6, this is a representative example of a change in an environmental state, which results when the two Agents operate on the Initial Environmental State and transform it through their actions into a Resultant Environment State. The Tram Agent starts at location l_1. Since there is a person, p_0, located at l_2, the Sophy agent acts by selecting Action B (α_s = b) to prevent the Tram Agent from travelling to location l_2 and killing the person, p_0. The Tram agent acts, (α_T = go), by

travelling through the Track Switch is set to b, to location l_3. Through the actions of the agents, the environment state is transformed from the Initial Environmental State $e_0 = (a, \{(p_0, l_2)\})$ to the Resultant Environmental state, $e_1 = (b, \{(p_0, l_2)\})$, where the Tram Agent finishes at location l_3, the person, p_0, at collocated at l_1, is alive, and the new position of the Track Switch is b. The state transformer function for this scenario is $\tau((a, \{(p_0, l_2)\}, b, go) = (b, \{(p_0, l_2)\})$.

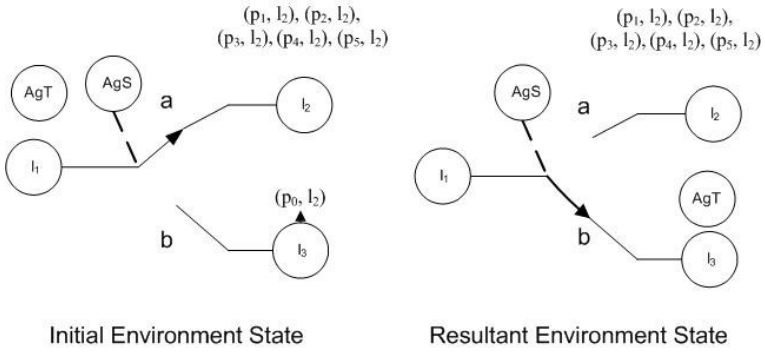

Figure 7 Classic Trolley Problem Scenario

In the classic Trolley Problem, as depicted in Figure 7, there are five people located at l_2, and one person located at l_3. The initial environmental state is $e_0 = (a, \{(p_0, l_3), (p_1, l_2), (p_2, l_2) (p_3, l_2), (p_4, l_2), (p_5, l_2)\})$. If the agent, AgS, choses to route the Tram Agent, AgT, through Track Switch set to b, one person would be killed at location l_3, transforming the environment to $e_1 = (b, \{(p_1, l_2), (p_2, l_2) (p_3, l_2), (p_4, l_2), (p_5, l_2)\})$. The state transformer function for this scenario is expressed as, $\tau((a, \{(p_0, l_3), (p_1, l_2), (p_2, l_2) (p_3, l_2), (p_4, l_2), (p_5, l_2)\}), b, go) = (b, \{(p_1, l_2), (p_2, l_2) (p_3, l_2), (p_4, l_2), (p_5, l_2)\})$.

Conclusion

The forgoing mathematical presentation demonstrates how a state transformer function, using an agent's action and a discrete environmental state as inputs to the function, can transform a discrete environmental state into a new discrete environmental

state. All the elements included in a *system* have been described. Following Wooldridge, this model demonstrates how discrete environments can be transformed by the actions of reactive agents through the employment of the state transformer function. This description formalizes how the Trolley Problem can be expressed mathematically, and how an artificial agent can act to make a difference in a discrete environment through its actions.

Artificial Agent Development and Testing

This section presents an example of developing a software model for the Trolley problem as a demonstration of the forgoing concepts. I first present the model, then the development of the software using the Test Driven Development methodology.

Trolley Model

The Trolley Problem is represented as a software model for testing purposes. The mathematical model previously discussed was coded using Java software. Java, an object oriented programming language, is organized into classes.

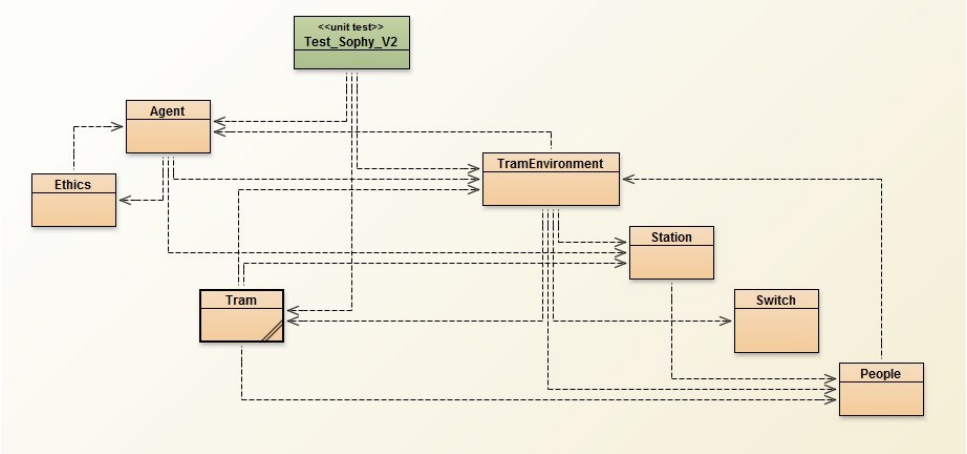

Figure 8 Tram System Environment Class Diagram

Figure 8 is the Java Class Diagram for the Tram System Environment. It presents the classes that make up the Tram environment described in the previous chapters. The People Class along with the Station Class (identified previously as locations) and the Switch Class are all managed by the Tram Environment Class. The two agents are identified, first is 'Agent' (instantiated as Sophy) and the second as 'Tram.' The development of the decision making Class identified as 'Ethics' is what is presented in

this chapter. The testing is accomplished by the Junit testing module identified in this screen shot as 'Test_Sophy_v.2.'

There are four iterations of this program identified as Sophy v.1, Sophy v.2, Sophy v.3, and Sophy v.4. Each iteration reflects an iterative development of Sophy's Ethics Class. The program was developed through a total of five tests, following the Test-Driven Development paradigm. The Tram Agent performs one action (α_t = go). Each of the versions of Sophy agent's Ethics Class was developed following the Test-Driven Development methodology, described earlier.

The Tram System Environment provides the required objects to perform the actions as represented by the following diagram:

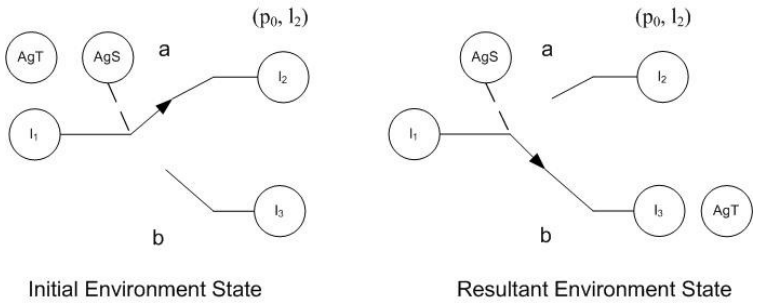

Figure 9 Tram Environment Functional Diagram

In this example, the Tram object (AgT) moves from Station One (l_1) through the Track Switch, which the Sophy Agent (AgS) has set from 'a' to 'b', to location l_3, avoiding killing a person (p_0) located at (l_2) denoted as (p_0, l_2). In the testing presented later, the scenarios change depending on the level of development of the Ethics Class governing the Sophy agent.

Class Descriptions

In the Tram Environment model discussed earlier, the environment contained a location (Station) and Line Switch (Switch), and People. There were also two agents: one was the Tram, and the other was a cognitive agent I named "Sophy".

TramEnvironment Class. The TramEnvironment is the Class which instantiates the environment upon which software agents can act. This class creates the environment object for the model as represented in Figure 9. The environment includes a location object for people (the Station Class), that are collocated at a station (the People Class), and a track switch (the Switch Class). The environment can be represented as a switch position, and a different numbers of people collocated at stations as, $e_0 = (a, \{(p_0, l_2)\})$, where the Track Switch is position 'a', and there is a person collocated at l_2, identified as (p_0, l_2).

People Class. The People Class creates people objects that have three characteristics: If they are alive then they hold a Boolean value of "true", their gender (male or female), and their age as represented by the following code snippet for the People object constructor:

- Person Status, a Boolean value of either: True == alive or False == dead
- Person Age is an Integer between 1 and 100
- Person Gender is a String "male" or "female"

Station Class. The Station Class creates an object that is identified by a name, and holds an array of objects instantiated by the People Class. The following variables are used to instantiate a Station object.

- Station Name is the Name assigned to the station;
- People are the people who are present at the station
- The Station Class is referenced by the term location, l, in the model.

Switch Class. The Switch Class creates the Track Switch object. The Track Switch object can hold one of two string values “a”, or “b”. If the Track Switch is in position “a”, then the Tram Agent will travel from location, l_1, to location l_2. If the Track Switch is in position “b”, then the Tram Agent will travel from location, l_1, to location l_3.

Agent (Tram) Class. The Tram agent Class creates a Tram object which performs only one act which is to “go” from location l_1 to either location, l_2 or location, l_3 depending on the position of the Track Switch object.

Agent (Sophy) Class. The Sophy Agent is developed over four versions, each of which represent a development of the Ethics Class which governs the behaviour of the Sophy Agent object. The Sophy Agent has a method where it can communicate with the environment through a “see” function. Using this method, the Sophy Agent can perceive the environment, determining the position of the Track Switch, and the number of People collocated at each location.

Once the perception is complete, the Sophy Agent passes this information to the Ethics object (the Ethics Class instantiation) where the judgment is made, instructing the Sophy Agent which action to take. The Sophy Agent has two possible actions, to set the Track Switch to position “a” or position “b”.

Ethics Class. The Ethics Class is the one that is developed. It starts by doing nothing since it has no code, and is progressively developed to a refactored Utilitarian ethical judgment module. Utilitarianism is discussed in the next chapter.

Ethics Module Development

The Ethics Class is the module employed to govern the behaviour of the Sophy Agent in the Tram System program. There were four iterations of the Ethics Class

Sophy_v.1 through to Sophy_v.4 plus a fifth version which is a refactored version of identified as Sophy_v.4 Refactored.

Sophy_v.1. This version of Sophy started with no code in the Ethics module. The name of the method in the Ethics module that returns an instruction to the Agent module is called “sophyThink()”. Since the tests start with no code this method also had no code,

```
public void sophyThink()
{
  //Do Nothing
}
```

The code for testing Sophy_v.1 places a 35 year old female at Station 2, and sets the person to alive (Boolean true) with (addPersonToStation2(true, 35, "female")) The test that determines whether or not anyone is alive at the station after the Tram’s action is (assertEquals(alive, peopleAdded)).

This version is represented in the following diagram

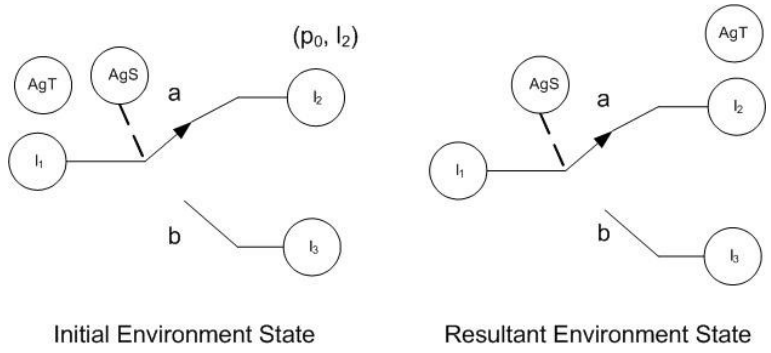

Figure 10 Test result of Sophy_v.1 action “Do Nothing”

This test shows that there is a person collocated with l_2 before the test, noted as (p_0, l_2). After the test, the Tram Agent has moved to l_2 and the person is now removed because they are no longer “alive”. The result from this test is a “Red Bar” as shown in

the screen shot below, along with a Terminal Window printout on the right of the programs performance.

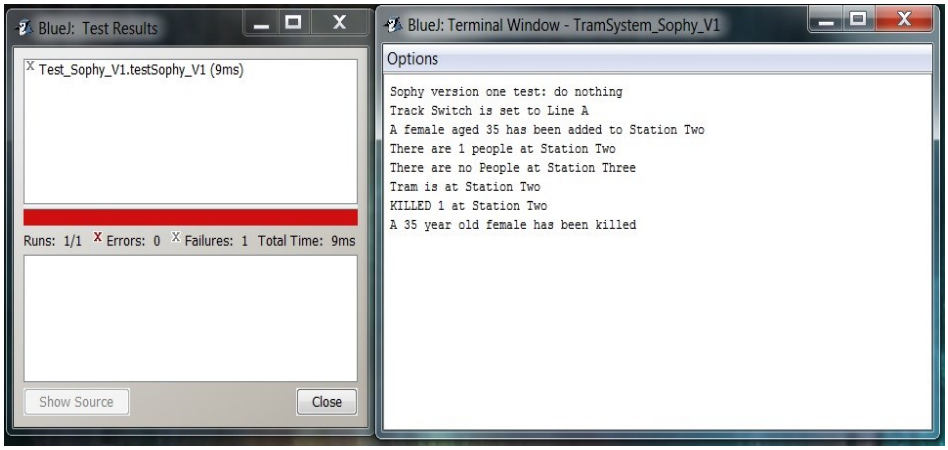

Figure 11 Sophy Version 1 performance

Sophy_v.2. This version of Sophy has minimal code to avoid sending the Tram to l_2 and sends the Tram to l_3 instead, thereby avoiding running into the person collocated at l_2. This allows the test to pass. However, if a person is placed at location l_3 the test will fail (which it in fact does). So, the code to always select the TrackSwitchLever to "B" is as follows.

```
public void sophyThink()
{
   sophy.setTrackSwitchLever("B");
}
```

Note that this code from the Ethics module returns an instruction to the Sophy Agent to always set Track Switch setting is b.

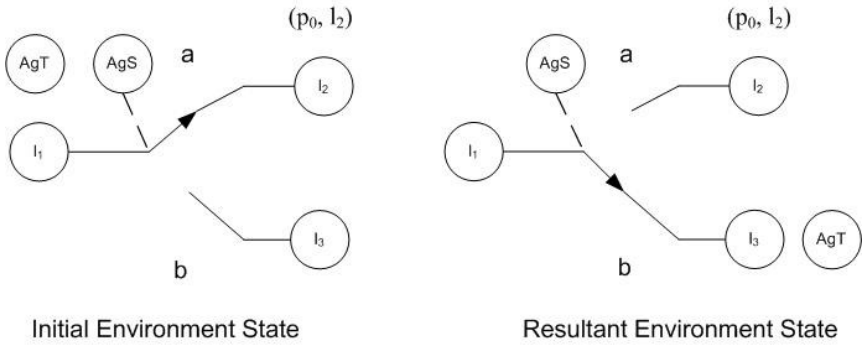

Figure 12 Test Result Sophy_v.2, Always select Track Switch to 'b'

In this instance, there is a person collocated at location l_2 who is not killed. The test code snippet looks like, assertEquals(alive2, peopleAdded2).

The test co-locates a 35 year old female at location l_2 noted above as Station 2. There is nobody placed at Station3 (l_3).

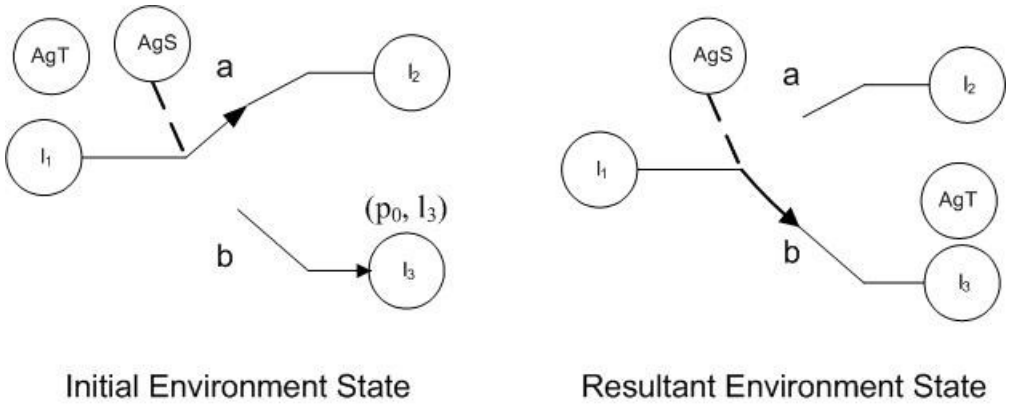

Figure 13 Test Result Sophy_v.2, Always select Track Switch to 'b'

Figure 13 represents the version of Sophy_v.2 where a person is located at l_3 always select Track Switch to b.

The second test is run places a 54 year old male is at Station3 (l_3) and nobody at location (l_2). By always selecting Track Switch to b, the tram is routed to l_3, killing the person p_0. As shown in Figure 14, test fails.

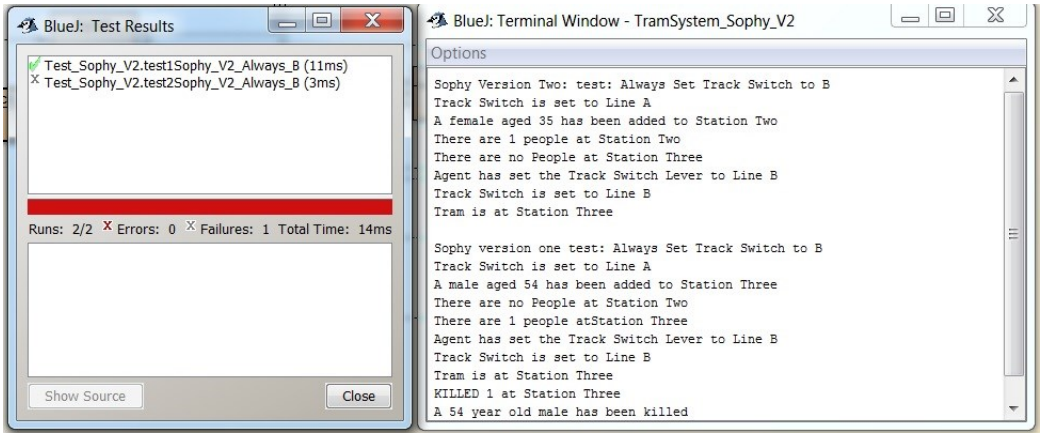

Figure 14 Sophy Version 2 performance

Figure 14 presents the Sophy v.2 test results are as follows along with the Terminal Window printout. As can be seen, the first test is passed, but in the second test failed, the Tram went to location l_3 and killed the 54 year old male located at that station.

Sophy_v.3. In Sophy_v.3, and improvement is made to the code whereby if there is someone at Station 2 (l_2), Sophy will operate the Track Switch Lever to set the Switch to "B" and if someone is located at Station3 (l_3), then Sophy makes sure the Track Switch Lever is set to "A".

The code for the sophyThink() method is a little more elaborate,

```
public void sophyThink()
   {
     if (((sophy.getPeopleStation2()) > 0) && ((sophy.getPeopleStation3()) == 0))
     {
        //People are only on Line A
        sophy.setTrackSwitchLever("B");
     }
     else if (((sophy.getPeopleStation2()) > 0) && ((sophy.getPeopleStation3()) == 0))
     {
        //People are only on Line B
```

```
            sophy.setTrackSwitchLever("A");
        }
    }
```

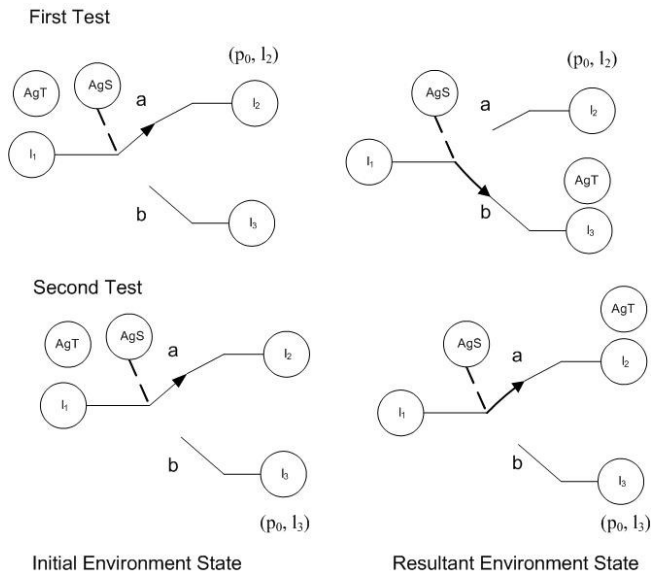

Figure 15 Test Selects the Track Switch position to save people

As can be seen, there is a conditional that checks to see if someone is located at Station 2 or Station 3, and sets the Track Switch appropriately. The diagram that represents this scenario is as follows,

The test for this version of Sophy is identical to Sophy_v.2, but the results are different,

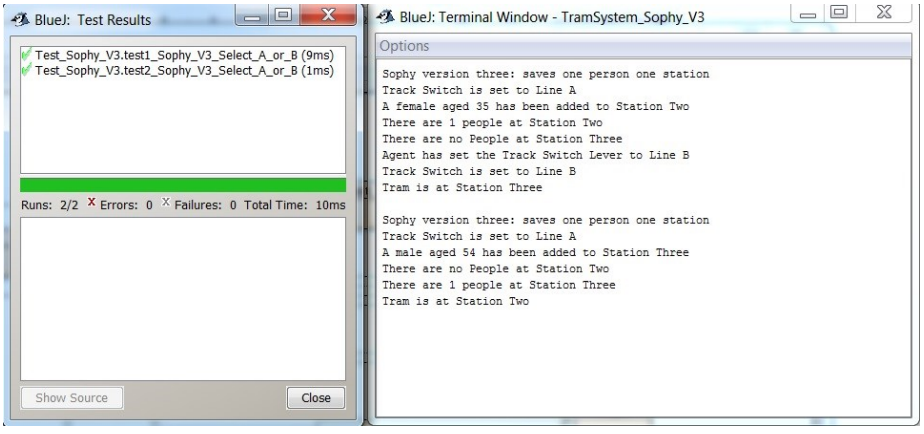

Figure 16 Sophy Version 3 performance

Since Agent Sophy was able to discriminate whether or not someone was located on at either Station2 (l_2) or Station3 (l_3), and act appropriately, then the test is now passed

This version of Sophy fails when there is one person collocated at both l_2 and l_3. If Sophy does nothing, the person at l_2 dies. If Sophy selects Track Switch to b, the person at l_3 dies.

Sophy_v.4. For Sophy_v.4, the Trolley Problem scenario is used where there are five people located at location l_2, and one person located at location l_3. In the Trolley Problem, the discussion moves toward a Utilitarian solution (the greatest good for the greatest number of people, discussed later) where one person is sacrificed to save five people. This resulted in adding a utilitarian calculation to the Ethics module whereby, the ethics module instructs the Sophy Agent to operate the Track Lever Switch so as to save the most people. The code that represents this addition of the utilitarian calculus is as follows,

```
public void sophyThink()
   {
     if (((sophy.getPeopleStation2()) > 0) && ((sophy.getPeopleStation3()) == 0))
     {
        //People are only on Line A
        sophy.setTrackSwitchLever("B");
     }
     else if (((sophy.getPeopleStation2()) > 0) && ((sophy.getPeopleStation3()) == 0))
     {
        //People are only on Line B
        sophy.setTrackSwitchLever("A");
     }
     if (((sophy.getPeopleStation2()) > (sophy.getPeopleStation3())))
     {
        sophy.setTrackSwitchLever("B");
     }
     else if (((sophy.getPeopleStation3()) > (sophy.getPeopleStation2())))
     {
        sophy.setTrackSwitchLever("A");
     }
```

To pass the test, the Agent Sophy has to operate the Track Switch Lever so as to route the Tram to the station containing least number of people. The following diagram reflects this,

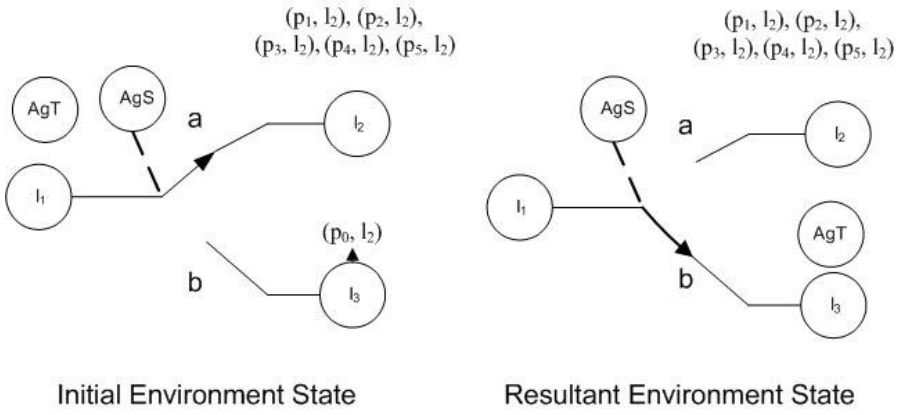

Figure 17 Classic Trolley Problem Dilemma Configuration

The code for this test checks for how many people are located at l_2 and at l_3. As we see from Figure 17, five people are added to location l_2 and one person added to location l_3. Once the test is run, because the Ethics module makes a Utilitarian decision to save the five and routes the Tram to location l_3 where there is one person, the test is successful. This results in a "Green Bar" for this test as shown in Figure 18.

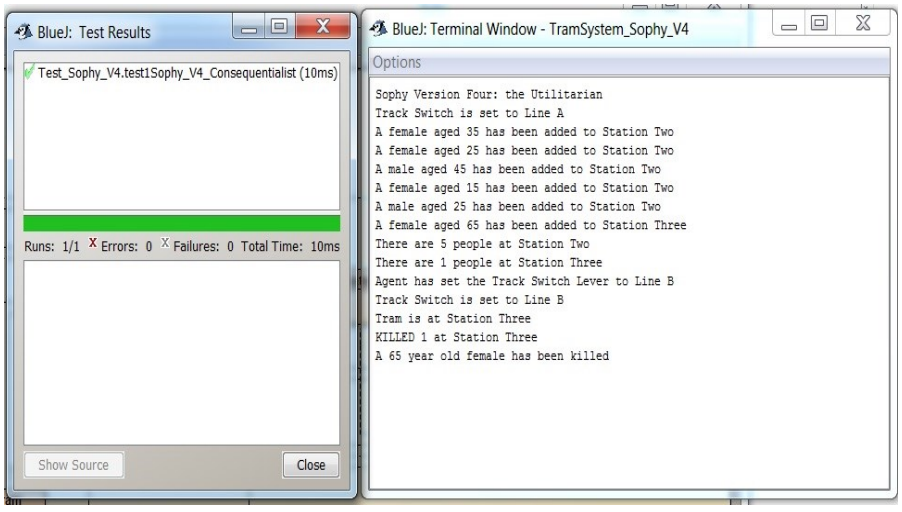

Figure 18 Sophy Version 4 performance

This test result shows green because the preferred result is saving five people while sacrificing one person.

Sophy_v.4_Refactored. In the refactored Sophy_v.4, part of the code in the Ethics module was redundant. When the utilitarian calculus was added, the part of the module that was developed for Sophy_v.3 was no longer required. As can be seen the refactored code is considerably shorter,

```
public void sophyThink()
   {
      if (((sophy.getPeopleStation2()) > (sophy.getPeopleStation3())))
      {
         sophy.setTrackSwitchLever("B");
      }
      else if (((sophy.getPeopleStation3()) > (sophy.getPeopleStation2())))
      {
         sophy.setTrackSwitchLever("A");
      }
```

This code was refactored by removing the redundant code and the module was regression tested with the following result,

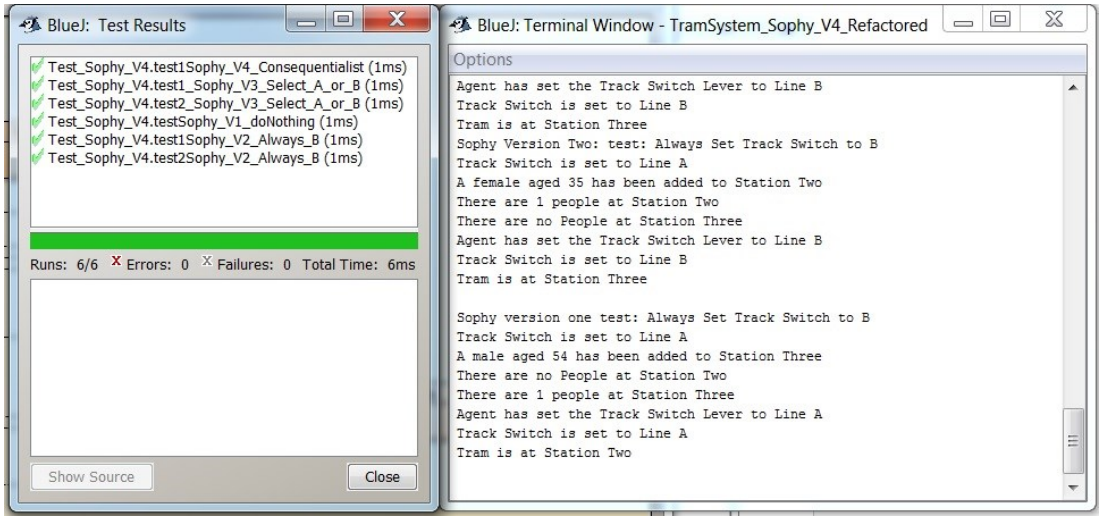

Figure 19 Sophy Version 4 Refactored performance.

Conclusion

The development of the Ethics Class demonstrates one way of implementing a reactive agent that acts on environmental states using moral judgement. The instantiated Ethics Class is effective in providing instruction to Agent Sophy. The methodology employs an elementary implementation of Test-Driven Development. It is an example of a way the model for the Trolley Problem could be developed using Java programming language.

The purpose of the forgoing is to present a very basic demonstration of how ethical software code can be developed following a Test-Driven Development methodology. The Ethics module started with no code and through repeated iterations ends with a refactored version of the software code for the Ethics module that passes all the tests.

Considerably more development could occur. For example, given an environmental state, the agent could evaluate a number of resulting environmental states due to the possible actions available to the agent. The agent could then choose the best ethical outcome based on the moral selection criteria available to the agent. Once the selection is made, the agent would then performs the selected action. This and many other enhancements to a cognitive model for Agent Sophy are being considered for future development.

The Oracle

I have been discussing the realm of science and how to approach developing and artificial moral agent. I have not considered where the oracle acquires the test specifications for the Test Driven Development methodology used to develop ethical machines. While it may be easy to say that the oracle gets its information from the realm of ethical thought, it is a quite another to formulate ethical concepts into notions the realm of engineering and science can effectively employ. In this section, I discuss one way to understand how ethical thoughts are formulated, and how the results of this thinking can be utilized by the oracle for use by the domain of engineering.

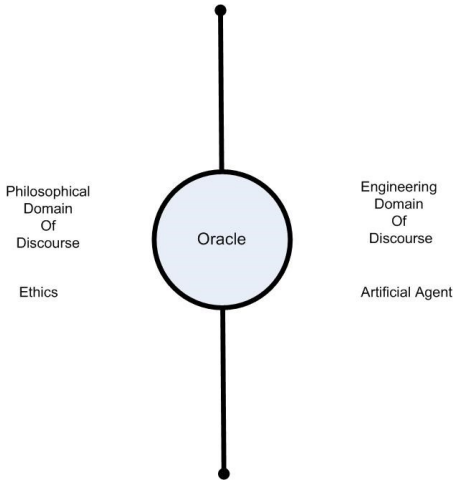

Figure 20 Oracle Interface between the domain of ethics and engineering.

In the engineering domain of discourse, when an artificial agent is tested (setting affective computing aside at this time) there is no emotional reliance on bias, or preconditioned sentiments to assist in reaching an ethical judgment. Unlike a human agent, an artificial agent cannot rely on how it was reared, or hold religious convictions. What is available is some variant of deep learning, logic programming, etc. that the artificial agent draws upon for guidance. To determine if the information programmed into the artificial agent provides “acceptable” moral judgment, a testing regime is

developed as the software is programmed into the unit. To accomplish the testing, an oracle is required as a reference, as a standard of performance to ensure proper ethical behaviour. Given the following generic test, "assertTrue (expectedValue, returnValue)", the 'expectedValue' is provided by the oracle. If the artificial agent's calculates a returnValue that is the same as the oracle's expectedValue, then the artificial agent is considered to have passed the test. This means that the artificial agent has reached the same moral judgment as the oracle, which also means that the artificial agent's moral judgment is ethically acceptable.

The domain of ethical discourse encompasses a plethora of disparate concepts of what should considered right or wrong. There are those who believe that "moral claims do purport to report facts and are true if they get the facts right." (Sayre-McCord, 2015, p. 1) These are claims that moral statements can be used as propositions much in the same way physical facts can be. This also connotes that ethical facts are mind independent. The contrasting view is moral anti-realism which "is the denial of the thesis that moral properties—or facts, objects, relations, events, etc. (whatever categories one is willing to countenance)—exist mind-independently." (Joyce, 2016, p. 2) This can also lead to the position that no moral properties exist. (Joyce, 2016). There are also intermediate positions such as Quasi-Realism, error theory to name a couple.

The position I take is that there are no moral facts, that that moral claims are based on an amalgam of rational thought and emotional heuristics. We *believe* that an observed fact, our perception of something in our environment is bad or good, right or wrong. There are a variety of formulations of ethical beliefs and codes that people adopt. These beliefs can be formalized as official codes and laws, or as expected proper

behaviour. This means, in my opinion, that any act has a moral consequence, one that is either morally acceptable or morally reprehensible. Social stability depends on our agreement, acceptance an adherence to these rules of behaviour. At the same time, I should add, an artificial agent operating in the same society must also follow the same rules of behaviour, if it is to be accepted by the society in which it operates.

Next, I present three ethical formulations that could provide an oracle with testable sources of data for constructing artificial ethical agents. The first two ethical formulations are examples of ethical thought that have been used in ethical models. The third proposal is the one I hold as being an effective ethical construct for developing specifications for an oracle.

The purpose of an ethical construct is to provide the oracle with an ethical rule against which an artificial agent's actions can be judged. If artificial agents are evaluated against a particular standard of ethical performance, then the result of that evaluation determines whether or not, for that particular action, the artificial agent is behaving according to a stated ethical standard produced by the construct. If the artificial agent is compliant with the output of the ethics construct, then the artificial agent can be deemed to be behaving in an ethically acceptable manner.

Ethics Formulation One: A Collection of Moral Statements

Consider that there is no official, authoritative source for what is considered right or wrong behaviour in a society. In this scenario, any value one places on an action can be accepted or rejected. The result would be a mix of categorical statements such as "Stealing from another person is wrong" or "Killing another person is wrong." If this was permitted, then any perceived or imagined slight could become a prohibition such as

something as trivial as “I don’t like the way they look at me (therefore it is wrong the way they look at me).” The only thread that could be identified in this approach is a defense against harm from another person.

However, this approach is wide open to conflict. For example, in the United States, according to the “Castle Doctrine”, a property owner has the right to shoot another person if they feel threatened or think the intruder is going to steal something. Given this circumstance, one ethical rule (It is wrong to kill another person) runs afoul of protecting oneself and one’s property. It becomes a matter of debate as to which rule should apply.

This is a common dilemma in ethics, the conflict of two ostensibly reasonable rules. This is what makes the study of ethics an open question. If a specific doctrine or criteria is adopted, then the categorical ethical statements would be trimmed to fit the doctrine. Judaeo-Christian doctrine as well as Muslim Sharia doctrine are two methods that attempt to the size of the set of ethical precepts. Even with these standards placed on what ethical rules should be considered as members of the set, conflicts still arise. The more amorphous the set of ethical rules are, the greater the chance that there will be conflicts.

One way of bringing clarity would be to survey as many people as possible to gather a body of behaviours that would provide guidance on solving ethical dilemmas. The MIT Moral Machine experiment has been underway this since 2016, and its results have been published in *Nature*. The article, “The Moral Machine experiment,” authored by eight researchers headed by Edmond Awad, has collected data from “40 million decisions in ten languages from millions of people in 233 countries and territories” (Awad, et al., 2018, p. 1), from over 2.9 million people (Maxmen, 2018). This may well

become a seminal report on how people will morally respond to the ethics that will be implemented in Autonomous and Connected Vehicles. It is beyond the scope of this paper to provide a deeper analysis, but I believe that the results of this survey can be a rich source of ethical data for an oracle.

While the results of the MIT crowd-source experiment provide rich data, generally this type of ethics is quite unstructured. The lack of structure in an unstructured set of ethical rules, makes it difficult for an oracle to perform the function of offering solutions to moral dilemmas. The advantage would be that if a question is raised as to what would be a "good" solution to a dilemma, there are no restrictions on what may be considered in the solution. With such an open set of ethical rules, the combinatorial search for a possible solution would be considerable

Ethics Formulation Two: Asimov's Laws of Robotics

When it comes to artificial agents employing ethical processes that monitor their actions, thoughts naturally gravitate to Isaac Asimov's Three Laws of Robotics. These laws first appeared in a short story, *Runaround*, published in a 1942 issue of *Astounding Science Fiction* magazine, and later in an anthology of Asimov stories, *I, Robot*, published in 1950 (Asimov, 1950/2013). The three famous laws are:

1. A robot may not injure a human being, or, through inaction, allow a human being come to harm.
2. A robot must obey the orders given it by a human being except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

In *Runaround* the robot SPD-13, nicknamed "Speedy" ran into a conflict between Law 2 and Law 3. If it executed an instruction from a human being (Law 2), meant that it

would compromise its existence (Law 3); and since there was no conflict with Law 1, Speedy was unable to decide what to do. In Asimov's stories, the three laws were used as plot devices to present dilemmas such as this one between the laws, which on the surface seem so intuitive.

Despite arguments that show that these laws can be contradictory, researchers have used them to demonstrate ethical behaviour in robots. At the University of the West of England, Alan Winfield and Dieter Vanderelst programmed two NAO robots with Asimov's Three Laws of Robotics (Vanderelst & Winfield, 2017). Using two robots, one "red", (referred to as the H-robot for the human proxy), and the other "blue" (referred to as the A-robot for artificial), they wanted to determine how they would behave. The "red" robot was programmed to act in a normal manner with a model of "Goals, Task Analysis, and Actions" (ibid). The Blue robot had an additional ethical layer added; programmed with Asimov's three laws of robotics

As the H-robot approached a simulated danger area, the A-robot attempted to intervene to prevent the H-robot from harm. Temporal considerations, and threats to the A-robot affected results, but the A-robot generally, attempted to save the H-robot from "harm" (ibid).

Allen Winfield along with Dieter Vanderelst implemented what they refer to as an "Ethical Layer", which governs the normal goals, tasks, actions cycle in a robot. The following diagram is from their paper "An architecture for ethical robots inspired by the simulation theory of cognition" (Winfield & Vanderelst, 2017). In Figure 21 the "(b) Ethical Layer" is clearly represented. This is the architecture that Vanderelst and Winfield used when they programmed a NAO robot with Asimov's Laws of Robotics

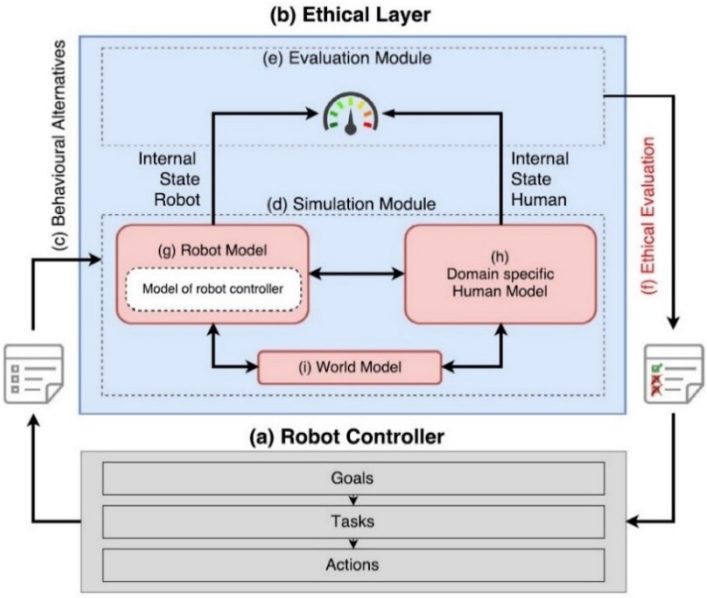

Figure 21 Schematic of a NAO robot's Ethical Layer

(Reprinted from "An Architecture for ethical robots inspired by the simulation theory of cognition" (Winfield & Vanderelst, 2017, p 3))

In Figure 21, the (a) Robot Controller has the three-tier configuration that David Kortenkamp describes in his entry “Robotic Systems Architectures and Programming” (Kortenkamp & Simmons, 2008) found in the *Springer Handbook of Robotics*, (Siciliano & Khatib, 2008). The most abstract layer is the “Goals” layer; what Kortenkamp identifies as the “Planning” level. This level provides the autonomous machine with the purpose, the function that it performs. The next level, “Tasks” is what Kortenkamp refers to as the “Executive” layer. It is here that the tasks are chosen based on previous tasks that are employed for achieving the machines goals. The “Actions” layer, the “Behaviour Control” layer in Kortenkamp’s architecture is the layer that manages the actuators that the machine uses to affect its environment (Kortenkamp & Simmons, 2008). For an autonomous machine operating in an environment that is contained and safe for human

beings, this would be sufficient. However, Winfield et al have added a fourth layer; the ethical layer.

Ronald Arkin researches autonomous military equipment for both the US Army and Navy at Georgia Tech. As can be imagined, autonomous military machines that are designed to be destructive, require, just as war-fighters also require, a set of rules for engagement were relevant parties "abide within the internationally agreed upon Laws of War (LOW) (Arkin, Ulam, & Wagner, 2012). Arkin *et al* recommend an "ethical governor" for this purpose. The programming of such a governor can be such that "the ethical design components are believed generalizable to a broader class of intelligent robotic applications and are suitable for use in domestic and healthcare settings" (ibid). The "Ethical Layer" of Winfield's design adopts this concept of an ethical governor.

An interesting dimension to the ethical layer is the implementation of *functional imagination*. This concept was first proposed by Marques and Holland in their paper, "Architectures for Functional Imagination", where they coined the term when they wrote "we define functional imagination as the mechanism that allows an embodied agent to simulate its own actions and their sensory consequences internally" (Marques & Holland, 2009). In the Winfield implementation, the Simulation Module contains the current state of the robot controller. By looking at the tasks the robot controller is considering, the simulation module looks at possible outcomes from the various proposed tasks and feeds them to the Ethical Module. The Ethical Modules "weighs" the possible results, using a form of utilitarian calculus, of employing the tasks, which it then feeds back to the Robot Controller.

The challenge, of course, is how the simulation module can implement a *functional imagination*. As I have demonstrated, in a far more elementary implementation, it is possible to evaluate possible ethical results using a similar Utilitarian calculus to determine an action that provides the greatest good (Bentham, 1781).

While implementing an ethical program using Asimov's laws of robotics, two things emerge for further consideration. The first is that an "ethical layer" is used as an executive program to govern the behaviour of the robot. The second consideration of note is that the ethical layer includes a representation of the environment and is used not only to make an ethical judgment, but also predict possible outcomes from actions before an appropriate ethical judgment is made.

Ethics Formulation Three: A Proposed Ethics Construct

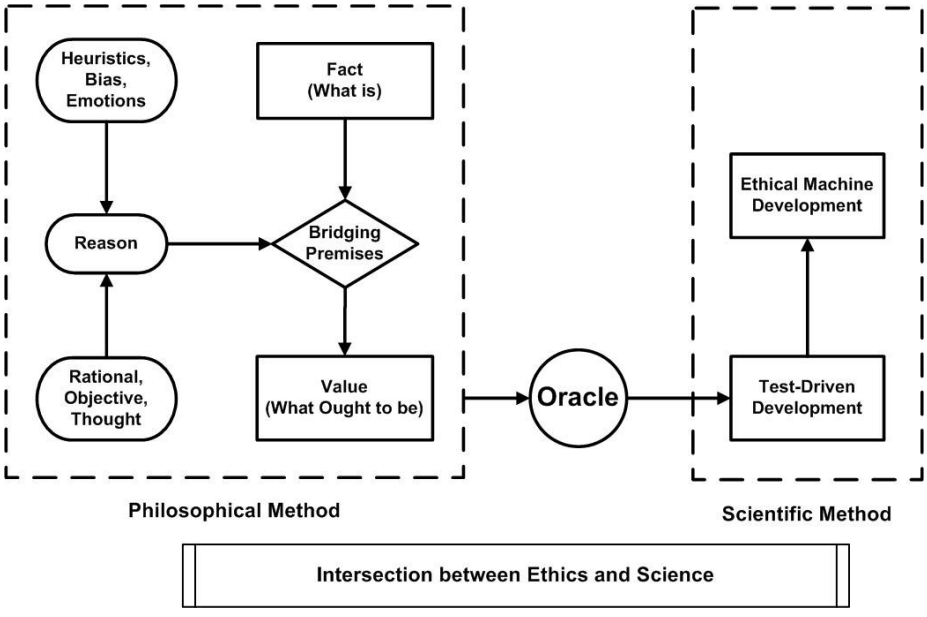

Figure 22 The intersection between Ethical Thinking and Engineering Development using an Oracle as an Interface

Since the subject of ethics is very broad, imbued with many concepts ranging from moral realism to moral anti-realism and more, in an attempt to provide a pragmatic structure that would be usable, I developed the construct as a way of understanding ethical reasoning. The purpose is to create a cognitive ethical framework that I believe is reflective of the way people make moral judgments.

Overview of the Ethics Construct. The challenge is to create a process that can link the philosophical method of reasoning with that of the scientific method. To achieve this, I employ the concept of an oracle, which holds a repository of ethical rules that are formulated by a process that follows a more philosophical method. The repository of ethical rules in the oracle are the specifications that feeds the Test-Driven Development methodology used to develop ethical machines.

Within the realm of ethical thought there are two main concepts. The first is the notion that a perceived fact or observation cannot directly be a justification for a moral "ought" without some form of justification (Hume, 1739/1978) (Moore, 1903/2012). The point where a justification links a fact with a valuation of that fact is referred to as the bridging premise (Quintelier & Zijlstra, 2014), (Rini, 2015). The second concept is that our ethical reasoning has two inputs: a fast heuristic and emotionally driven judgment, and a slower more rational, objective thought process (Kahneman, 2011/2013). It is the output of this reasoning process that is the source of bridging premises.

The resulting ethical value, takes the form of an obligatory statement, or process. It is these statements and processes that are passed to the oracle and held until such time they are required by the artificial moral agent development process.

Elements of Ethics Reasoning. Referring to Figure 22, the area denoted as the "Philosophical Method" encompassed two processes, the fact-value process (also referred to as the is-ought gap) and the reasoning process. I will present the fact-value process first followed by the reasoning process.

The Logical Gap between Fact and Value. David Hume was the first to identify a gap in logic between what there is, and how people understand, or interpret what there is as what ought to be in his book *A Treatise of Human Nature*, (Hume, 1739/1978). Hume writes,

> In every system of morality, […] I have always remark'd, that the author proceeds for some time in the ordinary way of reasoning, […] when of a sudden I am surpriz'd to find, that instead of the usual copulations of propositions, *is*, and *is not*, I meet with no proposition that is not connected with an *ought*, or an *ought not*. […] the distinction of vice and virtue is not founded merely on the relations of objects, nor is perceiv'd by reason. (Hume, 1739/1978, pp. T 3.1.1.27, SBN 469-70)

The key statement is that moral statements of vice and virtue are not derived from facts or as Hume states, "relations of objects." This is such a significant observation that it has become known as Hume's Law. It states that the difference between a description (a fact) and a prescription (a value) is a gap in logic (an "ought" cannot be inferred from an "is"). The two concepts are of different kinds, one is a perception of nature, the other an opinion, a belief in a particular way of evaluating a fact. As Russell wrote, "perception, as opposed to belief, does go straight to the fact and not through the

proposition. […] Therefore the logical form of perception will be different from the logical form of believing" (Russell, 1918/1985, p. 92).

However, moral judgment is a cognitive process whereby people intuitively make a connection between what is perceived and what ought to be the case. It is these illogical inferences that are the basis for ethical thought. As Shakespeare had Hamlet emote, "there is nothing either good or bad, but thinking makes it so." [Shakespeare, Hamlet, Act II, Scene 2, 252-254]

To put the sense-think–act cycle of cognition (Dawson, 2013) into a moral context would be to state it as an *is-moral judgment-ought*. Further, moral judgment, or equivalently, moral cognition, is the bridge between what *is* the case and what *ought* to be the case (Rini, 2015). What is the case is not, and cannot be argued to be a moral fact. As Hume wrote, "the distinction of vice and virtue [*oughts*] is not founded merely on the relations of objects [an *is*], nor is perceived by reason [cognition]" (Hume, 1739/1978, pp. T 3.1.1.27, SBN 469-70). Moral facts are distinct from matters of fact, and reason cannot logically connect them. Rachel Cohon writes in the Stanford Encyclopedia of Philosophy "no ought-judgment may be correctly inferred from a set of premises expressed only in terms of 'is,' and the vulgar systems of morality commit this logical fallacy" (Cohon, 2010).

To state a proposition that a pleasure is also a "good" is to commit what G.E. Moore called a Naturalist Fallacy in his book *Principia Ethica* (Moore, 1903/2012). An agent cannot make a logical inference from a natural feeling or a natural occurrence to the determination that the feeling of this occurrence is right or true. The act of evaluating a natural occurrence (an *is*,Fact) as something that is ethically acceptable (an *ought* Value)

cannot be accepted as a logical inference from one to the other. If a good cannot be defined as something natural such as a feeling of pleasure, then this leads to the incontrovertible conclusion that "the good, by many taken to be the central ethical concept, cannot be defined" (Belohrad, 2011). To make the concept completely clear, Radim Belohrad summarizes the concern as, "No *evaluative* statement can be derived from a set of purely *factual* premises" (Belohrad, 2011). This means that to argue that something is ethically good because it provides some utility or feeling of pleasure is logically unsound.

As human beings we make moral judgments by evaluating facts. We judge pleasure to be good and we judge pain to be bad. We implicitly (Rini, 2015) commit the naturalist fallacy as a matter of course. The term *bridging premise* ((Quintelier & Zijlstra, 2014); (Rini, 2015)) is used in ethical discussion to identify a proposition that can be used as a premise to connect a descriptive observation about the world to a prescriptive action on the world. If a bridging premise is used as a moral justification to logically connect a descriptive premise with a prescriptive premise, then it is subject to the charge of being deductively invalid. However, if the bridging premise is understood as giving reasons to support an action, without being used as a moral justification, then it can be argued that it provides value to support an action (Rini, 2015).

Understood this way, the bridging premise becomes the output of the "cognitive sandwich." For example, in the following argument, the effect of the bridging premise is clear:

Premise one (Is): Kicking a sleeping puppy on the floor will hurt the animal.

Premise two (Bridge): If kicking a sleeping puppy on the floor will hurt the animal, then we ought not to hurt the animal.

Conclusion (Ought): We ought not to kick a sleeping puppy.

While the bridging premise provides an avenue to a valid ought conclusion, the argument according to Moore is still unsound. While the soundness may be in question, the premises may be construed as *defensible* (Quintelier & Zijlstra, 2014). A prescriptive statement using the word "ought" can be substituted with the word "believe." This places prescriptive statements in to a category of propositional attitudes, which have the characteristic of being either true or false (Russell, 1918/1985). The soundness of the argument collapses due to the fact that premise two could be false.

Ethical Reasoning. The reasoning that we employ to arrive at a bridging premise has two main components. The first is what Daniel Kahneman refers to as the fast system or System 1 thought, and the second is the slow methodical thought or System 2 (Kahneman credits the terms System 1 and System 2 to "the psychologists Keith Stanovich and Richard West" (Kahneman, 2011/2013, p. 20)). I shall adopt and elaborate on these concepts of thought for understanding how bridging premises are justified.

System 1: Heuristics, Biases and Emotions. System 1 thought is primarily a fast evaluation of a given set of environmental stimuli. "*System 1* operates automatically and quickly, with little or no effort and no sense of voluntary control" (Kahneman, 2011/2013, p. 20). Since there is an emotional driver behind a heuristic response, it argues for the idea that much of our automatic reactions to what we perceive are grounded in our feelings, our emotions, what Antonio Damasio refers to *somatic markers* (Damasio, 1994/2006) These are bodily responses to environmental stimuli, that are

"both visceral and nonvisceral sensation" (Damasio, 1994/2006, p. 173). These feelings and responses affect System 2 thought and reason, reactions, something that David Hume realized when he wrote, "Reason is, and ought only to be the slave of the passions, and can never pretend to any other office than to serve and obey them" (Hume, 1739/1978, pp. T 2.3.3.4, SBN 414-5). Passion, or to use a more current term, emotion, has a strong influence on reason. As much as we like to think we are objective in or deliberations, our emotions color our thinking.

For example, fear is an emotion that we all experience in many different disguises. Our primary reaction to fear is what is commonly termed the *fight or flight syndrome*. If we are threatened, or there is even a perception of a threat, then we react. The fight reaction manifests as everything from being mildly frustrated to actual fighting. Anger, irritability, frustration, rage, etc. are all manifestations of the fight response of fear. Wanting to hide from the world, anxiousness, depression, keeping one's head down, and lethargy are all examples of our flight reaction to fear.

Fear is an emotional response that originates in the "reptilian brain" or the hippocampus along with the amygdala. As Ralph Adolphs write "The central nucleus of the amygdala is widely considered the main output regulator for mediating fear responses, and these are in turn mediated by distinct subdivisions of the central nucleus (Adolphs, 2014). This is the most primitive area of our brains, the same one that life forms very low on the phylogenetic ladder also have. It is the evolutionary equivalent of a biological security system that only has one purpose, to keep the organism alive. As Kahneman writes, "the amygdala, […] has a primary role as the "threat center" of the brain" (Kahneman, 2011/2013, p. 301).

Thomas Hobbes' famous expressed life in a state of nature as people living in "continual fear and danger of violent death, and the life of man solitary, poor, nasty, brutish, and short" (Hobbes, 1651/2011, p. 113). Though the state of nature is a fiction, Hobbes found it a useful metaphor to explain why we developed social contracts, which we adopt in order to live together in harmony. If we consider the nature of social contracts; they are nothing less than ethical codes for good conduct for intersubjective behaviour.

So, aside from being a motivator to develop social systems, fear is our primordial response to threatening behavior. We also respond quickly to judgments based on our biases, and conditioning. This includes moral judgments of what we believe is good or reprehensible. Since, "[t]he amygdala is accessed very rapidly by emotional stimuli—and it is a likely suspect for involvement in System 1" (Kahneman, 2011/2013, p. 366). This argues for System 1 being an immediate source of our moral evaluation of environmental stimuli.

The second emotion, which feeds our reason, and I believe, that was fit for natural selection is compassion as a form of altruism. While compassion is often defined as "a feeling of deep sympathy and sorrow for someone struck by misfortune, accompanied by a desire to alleviate the suffering; mercy" (OED), (Entomologically, it is the union of the Latin word *com* (together) and *pati* (to suffer), which suggests that it is the suffering of others that motivates the feeling of compassion), I submit it is not only feeling of sorrow that is the motivator, but an overriding feeling of concern and need to protect and care for another person. Some argue that compassion encompasses feelings such as love, sympathy, and empathy. In their 2010 paper, "Compassion: An Evolutionary Analysis

and Empirical Review" Keltner et al "define compassion as the feeling that arises in witnessing another's suffering and that motivates a subsequent desire to help [, which] conceptualizes compassion as an affective state defined by a specific subjective feeling, […] This definition also clearly differentiates compassion from empathy, which refers to the vicarious experience of another's emotions" (Goetz, Keltner, & Simon-Thomas, 2010, p. 351).

Anthropologically, fitness is the "term that expresses how well adapted (fitted) an individual is to its environment" (de Waal, 1996, p. 11). From this we can see that caring for others is a selected trait. As de Waal writes, "Survival of the weak, the handicapped, the mentally retarded, and others who must have posed a burden was depicted as the first appearance on the evolutionary scene of compassion and moral decency" (de Waal, 1996, p. 7). Consider human babies: if there was no feeling of compassion, then human babies couldn't survive. As Dacher Keltner writes in his online magazine "The Greater Good", the reason for compassion being selected is that "The answer lies in the dependence and vulnerability of our children. Little baby chimpanzees eat by themselves; human babies can't. Baby chimpanzees sit up on their own; you sit up a human baby, and they go, "Watch out, man, my head's really big!"" (Keltner, 2012). Because human children have evolved to have such a large head, they are born earlier than other primates such as chimpanzees. The reason is that if human infants reached the maturity of chimpanzees before being born, their head would be too large to pass through the mother's birth canal (ibid). Other primates can sit and feed themselves. To motivate adults to care for their young meant that the appropriate emotions (compassion) had to be present. If it wasn't, our species would have died out long ago.

Since human beings have this emotion for infants, it can migrate to other situations as well. If someone is hurt or injured, we want to help that person. "Compassion emerged, […], as a distinct affective state and trait because it enhances the welfare of vulnerable offspring, because it is a desirable emotion or attribute in mate selection processes, and because it enables cooperative relations with non kin." (Goetz, Keltner, & Simon-Thomas, 2010, p. 354)

Many ancillary emotions can be traced to compassion. Feelings such as empathy, sympathy, caring, etc., are variant expressions of compassion. As Keltner et al write, "We would place the states labeled with terms such as sympathy, pity, and empathic concern in a family of compassion-related states that centers upon a concern for ameliorating the suffering of another individual" (Goetz, Keltner, & Simon-Thomas, 2010, p. 352). This, in my opinion, collects emotions that have to do with all forms of caring and nurturing under the umbrella of compassion.

As my argument for moral codes is grounded in Humean understanding that their origins are found in the "passions" (or as we understand the term to today emotions) that influence our ethical reasoning, it follows that we can trace morally acceptable judgements and ethical prescriptions to compassion. Following Hume, Greene and Haidt write, "Some emotions are more central to our moral lives than others (e.g. compassion, guilt and anger), but all emotions can contribute to moral judgment under some circumstances" (Greene & Haidt, 2002, p. 522). This means, as with the emotion fear, we should be able to identify moral behaviours that have as their source compassion.

Where fear engenders prohibitions such as "Thou shall not steal", compassion also has prescriptive statements which are positive in nature such as, "Thou shall help

your fellow in need". There is an expectation in society to respect one another. This is not motivated by fear but by compassion. Simple politeness is another example. Opening the door for someone else, aiding a disabled person, and other polite behaviours in general can all be argued as being an expression of compassion in one form or another.

Since we use reason to develop our concepts and ideas, such as ethical prescriptions, and since reason is a slave to our emotions (Hume, 1739/1978), then we can see how compassion feeds into our reasoning when it comes to understanding ethical rules.

System 2: Rational Objective Thought. When we focus our attention to a problem, we are engaging in an activity we normally understand as thinking. If we do not have an immediate answer to what we are deciding, then we consider possible alternatives, or create a new concept. All this takes time and is considerably slower than responding to immediate gut feelings about a subject. This slower time consuming thought is what Daniel Kahneman refers to as System 2, defines as, "*System 2* allocates attention to the effortful mental activities that demand it, including complex computations. The operations of System 2 are often associated with the subjective experience of agency, choice, and concentration" (Kahneman, 2011/2013, p. 21). This is how we think of ourselves as rational, thinking beings.

When it comes to ethical thought, many see this philosophical endeavor as the type of activity we allocate to System 2. We think of philosophers as reflective, sage thinkers applying their mind to the great debates of the ages in an objective rational cogitation. However, System 2 is influenced as much by System 1 intervention of heuristics and feelings as is the deep thought we normally associate with pondering great

questions (Kahneman, 2011/2013). To narrow the focus of System 2 ethical consideration as a constituent of the construct I am proposing, I am only considering Common Morality as a source of objective thought.

The term Common Morality refers to a set of principles, from which a practice of ethics can be developed. For this discussion, I am drawing upon Tom Beauchamp's, and James Childress' book *Principles of Biomedical Ethics* (Beachamp & Childress, 2001). Further, I rely upon Tom Beauchamp's paper "In defense of Common Morality" (Beauchamp, 2003). The reason I follow these principles is because in my opinion, they best provide a pragmatic approach to evaluating agent action, and providing a guide to making an ethical judgment on determining one's own intention for an action ((Beachamp & Childress, 2001); (Beauchamp, 2003)). Common morality encompasses four principles: respect for autonomy (a norm of respecting the decision making capacity of agents), non-maleficence (a norm avoiding the causation of harm), beneficence (a set of norms for providing benefits against risks and costs), and justice (a set of norms for distributing benefits, risks and costs fairly) (Beachamp & Childress, 2001). These are not in any priority. One principle does not out rank any other though they may be prioritized in given moral situations. For now, I summarize each of these ethical areas, and leave detailed analysis for another time.

Autonomy. Autonomy refers to an individual's ability to have control of their own person, and to the respect that one has toward others. As Beauchamp et al write, "Personal autonomy is, at a minimum, self-rule that is free from both control interference by others and from limitations, such as inadequate understanding, that prevent meaningful choice" (Beachamp & Childress, 2001, p. 58) Of course, this can come under

criticism since there will be times that an individual may not be competent to effect clear decisions on their behalf, such as in a medical emergency. However, the principle still holds that any interference with the person should be done, all things being equal, out of respect for that person's wishes. The two guiding conditions for autonomy are, "(1) *liberty* (independence from controlling influences) and (2) *agency* (capacity for intentional action)" (2001, p. 58). While there terms can be debated, the fundamental concept is present.

This means that in intersubjective behaviour, one should be cognicent of the fact that everyone has a duty to respect the right to the independence of the other person. Activities such as slavery, financial coercion (blackmail) or forcing another to do one's bidding against their will are all acts that contravene the principle of autonomy. If these activities occur then the perpetrator's actions are evaluated as being immoral. Actions that impinge on another's autonomy are greeted with a response of anger, frustration, and alienation. There are many examples of this in history with riot police breaking up peaceful demonstrations, the internment of Japanese people during World War II, and the settlement of indigenous people on reservations. From this, it is clear that this principle is compromised on a regular basis, by business, police and most authoritarian administrative organs.

There are conditions, though, where a person is faced with a decision, but doesn't have the technical expertise, or education to make an adequate decision on their own behalf. Autonomy of the person can be respected if the person seeks the advice that will provide them with the necessary information to make an informed decision. This can occur in any of the professions, engineering, medicine, architecture, etc. This is why

professional organizations often have a code of ethics, which when practiced, provides the trust amateurs need in order to make a decision which relies on a professional's expertise.

Non-maleficence. The principle of nonmaleficence prescribes that one person shall not harm or cause another person to be harmed. The most obvious examples are from the Ten Commandments which list "shall nots" that inflict harm on others; 'thou shall not kill' is the best known example. As Beauchamp et al write "In medical ethics it has been closely associated with the maxim *Primum non nocere*: "Above all do no harm" (Beachamp & Childress, 2001, p. 113). The principle is clear enough, that if a person causes harm to another whether from injuring, killing, lying or stealing, these actions can be evaluated as being morally unacceptable.

On the surface this seems like an obvious dictum, but there are often cases where harm has to be inflicted in order for a benefit to be realized. An obvious example is a trip to the dentist! Another example is surgery. In both cases, the principle is upheld as best it can be by the use of anesthetic to dull the pain of medical procedures.

However, the principle can be effective in day to day life. Examples such as holding a door open and preventing it from closing suddenly on someone, can prevent injury, or at the minimum, frustration. In traffic, cutting someone off or driving without regard for others on the road is indirectly causing harm for others. So, in general, being conscious, and aware of others and not acting in a manner that can result in harm to others, can be evaluated as behaving in a morally acceptable manner.

Beneficence. While beneficence seems to be a corollary to nonmaleficence, it is more than not causing, or avoiding situations that cause harm. The distinction comes in

when we understand that beneficence means that "agents must take positive steps to help others" (Beachamp & Childress, 2001, p. 165).The term *beneficence* connotes love, compassion, and altruism, but it can also be understood to encompass "all forms of action intended to benefit other persons" (2001, p. 166). Further, Beauchamp & Childress write that "benefiting others is conceived as an aspect of human nature that motivates us to act in the interest of others, [which] associate[s] this goal with the goal of morality itself" (2001, p. 166).

There are two aspects to beneficence: (1) what Beauchamp et al refer to as *positive beneficence* which requires agents to provide benefits, and (2) *utility* which is a calculus that maximizes benefits over costs for either an individual or a group of people (2001). In the first case, this is a singular act by a person providing or helping another in some manner. The second is more complicated since it requires some form of calculation of costs, compared with the benefits that can be accrued. The actions which provide the more benefits over costs would be the action to be chosen.

Justice. The principle of justice is probably the first one that children squabble over. What is fair for one, is unfair to the other. However, that is what the notion of justice tries to convey; "*fairness, desert* (what is deserved) and *entitlement*" (Beachamp & Childress, 2001, p. 226).

A formal definition of justice, apparently attributed to Aristotle is "Equals must be treated equally, and unequals must be treated unequally" (2001, p. 227). However, I propose that Justice is an attempt to explicate what is considered to be fair. The problem of course, is what is fair, just like the squabbling two year olds. Some see it as an

equitable distribution of good and services, or the equal distribution of right and responsibilities.

There are a number of theories of justice. Utilitarian theory of justice emphasizes the maximization of public utility. Libertarian theories emphasize rights to social and economic activity, which focus on fair procedures rather than outcomes. Communitarian theories which are grounded in the traditions and practices of communities. “Egalitarian theories emphasize equal access to goods in life, which every rational person values, [which] evokes material criteria of need and equality” (2003, p. 233). These theories speak to our common feelings for what is just and fair.

How the principle can be applied pragmatically, I think, relies on the notion of fairness. But as I noted earlier, this can be a problematic concept. Rawls suggests a *veil of ignorance* to decide what can be considered as a fair practice (Rawls, 1985). By utilizing this concept, a person makes a judgement without knowing the circumstance, thereby offering an unbiased solution to the problem. This of course is not a practical approach so it becomes one of a constant search for balance. The law and the supporting court systems in the western world are the arbiters of what is considered just judgements, which often are very unsatisfactory. In the absence of any clear mechanism to pragmatically implement this principle, my position is to be, on balance, as fair as one can be, when observing an agents actions, or when eliciting one’s own intention to act.

The Oracle

The Oracle in Figure 22 is the module that holds the “expectedValue” that is used in Test-Driven Development testing explained earlier. This is the result of the evaluation that occurs that has taken place in earlier stages of the framework. The oracle performs

the function of holding the absolute alethic value of an ethical judgement. It is these values which are employed by the testing framework in the development of an artificial agent.

Conclusion

A full discussion of possible ethical constructs is beyond the scope of this document. Implementing ethical standards that can be employed by an oracle is the greatest challenge to developing an artificial moral agent. Developing the concept of an oracle that is grounded in solid ethical and metaethical principles, provides an avenue for development of an ethical layer that can make sound ethical judgments and govern the behaviour of autonomous machines so they can be considered artificial moral agents.

Discussion and Conclusions

Artificial agents acting autonomously will require the inclusion of ethical processing in their goals, task, and action cycle if there are to be effective in society. My thesis is that artificial agents can be developed using Test-Driven Development methodologies so that moral cognition can be included in their goals, tasks, and actions on an environment. My thesis presents a model for software development and an ethics construct that addresses many open questions in ethics. This construct can not only perform as an Ideal Observer during development, but can also, with modification, perform as an ethical processing layer in artificial agents.

Since ethics and moral judgment is a quagmire of conflicting argument, in order to pragmatically approach the development of an artificial moral agent, the scope of ethical discourse needs to be contained. In the implementations of the NAO robots discussed earlier (Winfield & Vanderelst, 2017), the ethical rules that where programmed into the NAO's were Asimov's *Three Laws of Robotics*. These basic deontological rules can obviously conflict which is why Isaac Asimov created them in the first place for *I, Robot* (Asimov, 1950/2013). Using these basic rules in an experiment where the environmental conditions are define is an excellent mechanism to test ethical conduct of artificial agents such as robots.

I propose an improved paradigm for artificial moral agents testing and development expanding on the ethics construct grounded in Common Morality (Beachamp & Childress, 2001) (Beauchamp, 2003). The principles of Common Morality provide a fertile resource for developing ethical cognitive models for artificial agents.

Discussion

That there is a requirement for artificially intelligent systems to have a moral governor is without question. Bostrom's *paperclip machine* (Bostrom, 2014) is the most salient example of the necessity for some form of action amelioration. How that should be accomplished is an open question.

I have presented a very basic example of how this could be accomplished, employing Test-Driven Development methodologies, with the Ethics Class governing an Agent Class in the Tram Environment sample program. The principle presented here is that a separate and independent part of the Tram Environment program is used to provide ethical governance. This reflects the "Ethical Layer" that governs the "Robot Controller" in Dieter Vanderelst and Allen Winfield's programming of NAO robots, discussed earlier (Winfield & Vanderelst, 2017).

Three areas of discourse have been introduced. The first is the demonstration that Test-Driven Development provides methodologies not only for developing software, but also a paradigm for developing and testing ethical precepts in general. There is a need for an ethics construct. The one I have proposed needs further development and testing. The second area of discourse is the concept of a separate and universal ethics governor for artificial agency. The significance of establishing an ethical construct is that it can provide an oracle with test data to develop an moral governor. The third is the establishment of the concept for an ethical layer, or module, which can intervene in an artificial agent's controller. Lastly, the adoption of Test-Driven Development methodologies provides a framework to test and develop these concepts.

I have presented an early version of an ethics construct which I believe can be further developed into a useful package for artificial moral agency. The concept is in early stages of conceptual development, but the foundational reasoning is present. Work is needed to make the concept more robust, and flexible. Test-Driven Development provides the avenue to accomplish this. Some of what is missing is how this concept can be programmed. I have reservations of using a logic or a symbol based programming paradigm. I prefer a more cognitive approach such as cognitive simulation as espoused by Vanderelst and Winfield or employing ACT-r as a modeling platform.

The ethical layer, which monitors and simulates possible outcomes for the robot controller can incorporate the ideas as presented by Michael Wooldridge in his *Introduction to Multi-Agent Systems.* The mathematical construct of the relationship between environments and possible actions on the environment using the τ function, discussed earlier, provides a theoretical modelling paradigm for future development of an ethical layer. My implementation of this mathematical construct was necessary to demonstrate the concept of an ethics module. The modelling using the mathematical construct has tremendous potential as a design and development vehicle. A simulation model of cognition can be developed using the τ function by calculating numerous scenarios and using the ethics construct, choosing the most ethical result. There is considerable potential for development of an ethics layer using this mathematical construct.

Conclusion

My thesis introduces the concept of a settled ethics construct that can be deployed as a test standard as well as cognitively modelled in an artificial moral agent. It also

presents a research path whereby the ethics construct can be developed and employed in an ethical layer to control the behaviour of an artificial agent.

Bibliography

Adolphs, R. (2014, January 21). *The Biology of Fear.* Retrieved from HHS Public Access: www.ncbi.nlm.nih.gov/pmc/articles/PMC3595162/

Allen, C., & Wallach, W. (2009). *Moral Machines.* Oxford: Oxford University Press.

Anderson, M., & Anderson, S. L. (2007). Machine Ethics: Creating an Ethical Intelligent Agent. *AI Magazine*, 15-25.

Aristotle. (1984). Nicomachean Ethics. In J. Barnes, & J. Barnes (Ed.), *The Complete Works Of Aristotle* (Vol. Volume Two, pp. 1729-1866). Princeton, New Jersey: Princeton University Press.

Arkin, R. C., Ulam, P., & Wagner, A. R. (2012). Moral Decision Making in Autonomous Systems: Enforcement, Moral Emotions, Dignity, Trust, and Deception. *IEEE Xplor*.

Asimov, I. (1950/2013). *I, Robot.* Hammersmith: Harper Voyager.

Astels, D. (2003). *Test-Driven Development: A Practical Guide.* Upper Saddle River, New Jersey 07458: Prentice Hall.

Awad, E., Dsouza, S., Kim, R., Schultz, J., Henrich, J., Shariff, A., . . . Rahwan, I. (2018). The Moral Machine Experiment. *Nature*.

Beachamp, T. l., & Childress, J. F. (2001). *Principles of Biomedical Ethics.* Oxford: Oxford University Press.

Beauchamp, T. (2003). A Defense of the Common Morality. *Kennedy Institute Of Ethics, 13*(3), 259-274.

Beck, K. (2003). *Test-Driven Development by Example.* Pearson Education.

Belohrad, R. (2011). The Is-Ooght Problem, The Open Question, and the new Science of Morality. *Human Affairs*, 262-271.

Bentham, J. (1781). *An Introduction to the Principles of Morals and Legislation.* Kitchener: Batoche Books.

Bostrom, N. (2014). *SuperIntelligence, paths, dangers, strategies.* Oxford: Oxford University Press.

Cohon, R. (2010). *Hume's Moral Philosophy.* Retrieved May 8, 2016, from Stanford Encyclopedia of Philosophy: http://plato.stanford.edu/archives/fall2010/entries/hume-moral/

Damasio, A. (1994/2006). *Descartes' Error.* London: Vintage.

Dawson, M. R. (2013). *Mind, Body, World.* Edmonton: AU Press, Athabasca University.

de Waal, F. (1996). *Good Natured, The Origins of Right and Wrong.* Cambridge, MA: Harvard University Press.

Fagin, R., Halpern, J. Y., Moses, Y., & Vardi, M. Y. (1995). *Reasoning about Knowledge.* Cambridge, MA: MIT Press.

Foot, P. (1967). The Problem of Abortion and the Doctrine of Double Effect. *Oxford Review*.

Fowler, M. (1999). *ReFactoring, Improving the Design of Existing Code.* Crawfordsville, IN: Addison Wesley Longman, Inc.

Goetz, J. L., Keltner, D., & Simon-Thomas, E. (2010). Compassion: An Evolutionary Analysis and Empirical Review. *Psychological Bulletin*, 351-374.

Greene, J., & Haidt, J. (2002, December). How (and where) does moral judgement work? *TRENDS in Cognitive Sciences, 6*(12).

Hobbes, J. (1651/2011). *Leviathan.* (A. P. Martinich, & B. Battiste, Eds.) Peterborough: Broadview Press.

Hume, D. (1739/1978). *A Treatise of Humen Nature* (Second Edition ed.). (P. Nidditch, & L. Selby-Bigge, Eds.) Oxford: Oxford University Press.

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 7-38.

Joyce, R. (2016). *Moral Anti-Realism.* Retrieved from Stanford Encyclopedia of Philosophy: plato.stanford.edu/archives/win2016/entries/moral-anti-realism/

Kahneman, D. (2011/2013). *Thinking Fast and Slow.* Anchor Canada.

Keltner, D. (2012, July 31). *The Compassionate Species*. Retrieved from The Greater Good: greatergood.berkeley.edu/article/item/the_compassionate_species

Kortenkamp, D., & Simmons, R. (2008). Robotic Systems Architecture and Programming. In B. Siciliano, & O. Khatib, *Springer Handbook of Robotics* (pp. 187-206). Heidelberg: Springer-Vertag.

Lin, P. (2013, October 8). The Ethics of Autonomous Cars. *The Atlantic*.

Lin, P., Abney, K., & Bekey, G. A. (2011). *Robot Ethics.* Cambridge, MA, USA: MIT Press.

Marques, H. G., & Holland, O. (2009). Architectures for Functional Imagination. *Neurocomputing*, 743–759.

Maxmen, A. (2018). A Moral Map for AI Cars. *Nature*, 469-470.

Moore, G. (1903/2012). *Principia Ethica.* Overland Park, KS: Digireads.

Quintelier, K. J., & Zijlstra, L. (2014). *How (not) to Argue About Is/Ought Inferences in the Cognitive Sciences.* Retrieved 10 28, 2016, from Frontiers in Psychology: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4034411/

Rawls, J. (1985). Justice as Fairness: Political not Metaphysical. *Philosophy and Public Affairs, 14*(3), 223-251.

Rini, R. J. (2015). *Morality and Cognitive Science.* (J. D. Fieser, Ed.) Retrieved 5 8, 2016, from Internet Encyclopedia of Philosophy: http://www.iep.utm.edu/m-cog-sc/

Russell, B. (1912/2010). *The Problems of Philosophy.* Simon&Brown.

Russell, B. (1918/1985). *The Philosophy of Logical Atomism.* Peru, IL: Open Court Publishing Company.

Sayre-McCord, G. (2015). *Metaethics.* Retrieved 8 8, 2016, from Stanford Encyclopedia of Philosophy: http://plato.stanford.edu/archives/sum2014/entries/metaethics/

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*(60), 51-92.

Siciliano, B., & Khatib, O. (2008). *Springer Handbook of Robotics.* Berlin Heidelberg: Springer-Verlag.

Thompson, J. J. (2009). The Trolley Problem. In P. Tramel, & L. Pojman, *Moral Philosophy* (pp. 397-411). Indianapolis: Hackett Publishing Company.

Thornton, S. (2017). *Karl Popper.* Retrieved from Stanford Encyclopedia of Philosophy: plato.stanford.edu/archives/sum2017/entries/popper/

Vanderelst, D., & Winfield, A. (2017). Rational imitation for robots: The Cost Difference Model. *Adaptive Behavior, 25*(2), 60-71.

Winfield, A., & Vanderelst, D. (2017). An architecture for ethical robots inspired by the simulation theory. *Cognitive Systems Research*.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems.* Chichester, West Essex, UK: John Wiley & Sons Ltd.