Chapter 1

Robots with Internal Models: A Route to Self-Aware and Hence Safer Robots

Alan F.T. Winfield
University of the West of England, Bristol, UK

1.1 Introduction

The aim of this chapter is to set out the case for building robots with internal models as a possible route toward achieving a level of functional self-awareness that would usefully extend the capabilities of autonomous robots. The chapter argues that these capabilities will lead to enhanced safety – especially in physical human robot interaction (pHRI) – and, perhaps also, toward ethical behaviour in autonomous robots. Indeed, the chapter will advance the argument that safe and ethical autonomous robots may not be achievable at all without mechanisms for self-awareness.

Importantly, the ideas and mechanisms proposed in this chapter are intended to be realisable with current and near-future technology, i.e. using conventional computing platforms embedded within existing or buildable robot bodies, with existing devices for sensing and actuation. Thus, this chapter is primarily about how we might *engineer* practical self-awareness, for safer (and possibly ethical) robots in the near-term. This chapter will be less concerned with philosophical questions such as whether, or not, such robots are *really* self-aware, although we will touch upon the question of what behaviour might, if exhibited, be argued as evidence for *as if* self-awareness.

1.2 Internal Models and Self-Awareness

An internal model is a mechanism for internally representing both the system itself and its current environment. An example of a robot with an internal model is a robot with an embedded *simulation* of itself *and* its currently perceived environment. A robot with such an internal model has, potentially, a mechanism for generating and testing *what-if* hypotheses:

(1) *what if* I carry out action x? and, ...
(2) ... of several possible next actions x_i, *which* should I choose?

Holland writes: "an internal model allows a system to look ahead to the future consequences of current actions, without actually committing itself to those actions" (Holland, 1992, p25). This leads to the idea of an internal model as a *consequence engine* – a mechanism for estimating the consequences of actions. Dennett, in his book *Darwin's Dangerous Idea* (1995), develops the same idea in what he calls the *Tower of Generate-and-Test*; a conceptual model for the evolution of intelligence that has become known as Dennett's Tower. Dennett's tower is a set of conceptual creatures each one of which is successively more capable of reacting to (and hence surviving in) the world through having more sophisticated strategies for 'generating and testing' hypotheses about how to react.

Dennett's tower starts with Darwinian creatures; these have only natural selection as the generate and test mechanism, so mutation and selection is the only way that Darwinian creatures can adapt – individuals cannot. One the second floor are Skinnerian creatures, which can learn, but only by generating and physically testing all different possible actions, then reinforcing the successful behaviour. The third floor of Dennett's tower contains Popperian creatures, which have the ability to *internally model* the possible actions so that some (the bad ones) are discarded before they are tried out for real. A robot with an internal model, capable of generating and testing *what-if* hypotheses, would thus be an example of a Popperian creature within Dennett's scheme.

The use of internal models within control systems is well established, but these are typically mathematical models of the plant (system to be controlled). Typically a set of first-order linear differential equations models the plant, and these allow the design of controllers able to cope with reasonably well defined uncertainties; methods also exist to extend the approach to cover non-linear plant (Isidori *et al.*, 2003). In such internal-model based control the environment is not modelled explicitly – only certain exogenous disturbances are included in the model. This contrasts with the internal simulation approach of this chapter which models both the plant (in our case a robot) and its operational environment.

In the field of cognitive robots specifically addressing the problem of machine consciousness (Holland, 2003), the idea of embedding a simulator in a robot has emerged in recent years. Such a simulation allows a robot to try out (or 'imagine') alternative sequences of motor actions, to find the sequence that best achieves the goal (for instance, picking up an object), before then executing that sequence for real. Feedback from the real-world actions might also be used to calibrate the robot's internal model. The robot's embodied simulation thus adapts to the body's dynamics, and provides the robot with what [Marques and Holland (2009)] call a 'functional imagination'.

[Bongard *et al.* (2006)] describe a 4-legged starfish like robot that makes use of explicit internal simulation, both to enable the robot to learn its own body morphology and control, and notably allow the robot to recover from physical damage by learning the new morphology following the damage. The internal model of Bongard *et al.* models only the robot, not its environment. In contrast [Vaughan and Zuluaga (2006)] demonstrate self-simulation of both a robot and its environment in order to allow a robot to plan navigation tasks with incomplete self-knowledge; although making no claims to self-awareness their approach provides perhaps the first experimental proof-of-concept of a robot using self-modelling to anticipate and hence avoid unsafe actions.

[Zagal *et al.* (2009)] describe self-modelling using internal simulation in humanoid soccer robots; in what they call a 'back-to-reality' algorithm, behaviours adapted and tested in simulation are transferred to the real robot. In a similar approach, but within the context of evolutionary swarm robotics [O'Dowd *et al.* (2011)] describe simple wheeled mobile robots which embed within each robot a simulator for both the robot and its environment; a genetic algorithm is used to evolve a new robot controller which then replaces the 'live' robot controller about once every minute.

Does having an internal model make a robot self-aware? The answer to this question depends of course on what we mean by 'self-aware'. But, in some straightforward sense, if a robot has an internal model of itself, then that model accounts for the *self* in self-aware. More difficult to justify is any claim to *awareness*, since this depends not just on having an internal model, but what the robot does with that model. Self-awareness is a *property* that needs to be demonstrated by behaviours, in particular behavioural responses to novel situations that have neither been preprogrammed, evolved or previously learned[1]. Moreover, those behaviours must in some way flow from the robot's perception of its environment, including its position in that environment – and the (relative) position of other agents, and its assessment of the possible consequences of both its own actions and those of other actors in that environment.

1.3 Internal-Model based Architecture for Robot Safety

Simulation technology is now sufficiently well developed to provide a practical basis for implementing the kind of internal model required to test *what-if* hypotheses, outlined above. In robotics advanced physics and sensor based simulation tools are commonly used to test and develop, even evolve, robot control algorithms before they are tested in real hardware. Examples of robot simulators include Webots (Michel, 2004) and Player-Stage (Vaughan and Gerkey, 2007). While using simulation tools roboticists are well aware of the dangers in making claims about algorithms tested only in simulation. The term reality-gap is used as shorthand for the gap between the performance of real sensors and actuators and their approximated and idealised versions, in simulation (Jacobi *et al.*, 1995). Furthermore, there is an emerging science of simulation, aiming for principled approaches to simulation tools and their use (Stepney *et al.*, 2011).

Fig. 1.1 proposes an architecture for a robot with an internal model which is used to test and evaluate the consequences of the robot's next possible actions. The machinery for modelling next actions is relatively independent of the robot's controller; the robot is capable of working normally without that machinery, albeit without the ability to generate and test *what-if* hypotheses. The *what-if* processes are not in the robot's main control loop, but instead run in parallel to moderate the Robot Controller's normal action selection if necessary acting, in effect, as a 'safety governor'.

At the heart of the architecture is the Internal Model (IM). The IM is initialised from the Object Tracker-Localiser, and loops through all possible next actions; these next actions are generated within the Robot Controller (RC) and transferred to the mirror RC within the IM (for clarity this data flow is omitted from Fig. 1.1). For each candidate action the IM simulates the robot executing that action, and generates a set of model outputs

[1]It should be noted that there may be other characteristics of self-awareness, including some that are externally unobservable.

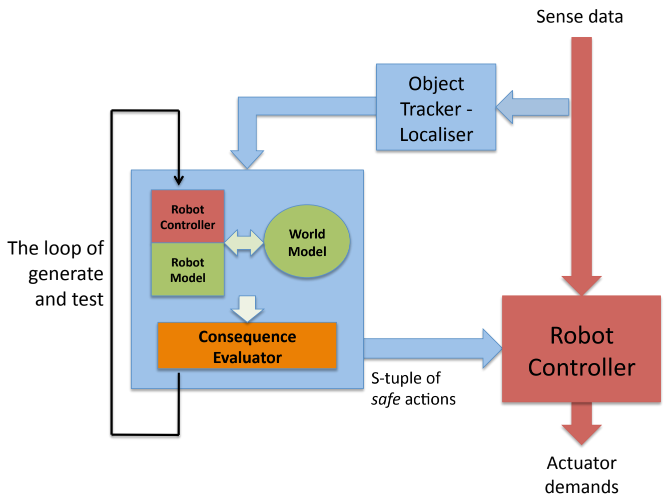

Fig. 1.1 A Self-Aware Architecture for Safe Action-Selection. The Robot Control data flows are shown in red; the Internal Model data flows in blue.

ready for evaluation by the Consequence Evaluator. The Internal Model and Consequence Evaluator loop through each possible next action; this is the Generate-and-Test loop. Only when the complete set of next possible actions has been tested does the Consequence Evaluator send, to the Robot Controller, actions it assesses to be safe. These processes are explained in more detail below.

1.3.1 *The Internal Model*

The Internal Model shown in Fig. 1.1 is a simulator which must incorporate both a World Model and a Robot Model. The World Model (WM) is a model of the robot's environment; including the terrain across which the robot must move (if it's a mobile robot) and the other objects the robot might encounter. Those objects might be static obstacles (i.e. walls) or hazards (i.e. holes in the ground), or dynamic objects. The dynamic objects could be moving obstacles or actors with which our robot must interact; these could be other robots or, if we are concerned with human-robot interaction, human(s). For many of the applications we might envisage the WM will also need to model real-world physics, so that for instance the inertia of moving objects is accounted for in collisions. The Robot Model (RM) is a model of our 'self-aware' robot and, as is commonplace in robot simulators, this will model the robot's sensors and actuators. Importantly the RM is controlled by the same Robot Controller as the real robot, so that it will act and react in the same way as the real robot. Thus the Robot Controller in the Internal Model mirrors the real robot's Controller - as shown in Fig. 1.1. All of these components are normally present in current technology robot simulators. However, in order to make use of such a simulator as our Internal Model we need the following additional capabilities.

(1) It must be capable of being initialised so that objects in the WM, and the state and disposition of the RM within the WM, can be matched to the real world environment and the location and disposition of the real robot in that environment. The initialisation data will be supplied by the Object Tracker – Localizer shown in Fig. 1.1.;
(2) It must be capable of being run with a given RM action, for a given simulated time period then halted and re-initialised to the same state, then run again for each of the robot's next possible actions;
(3) The final state of the simulator at the end of each of these fixed-time runs must be captured and suitably coded, then supplied to the Consequence Evaluator.

1.3.2 ***The Consequence Evaluator***

The purpose of the Consequence Evaluator (CE) is to compare the outputs of the IM for each of the robot's next possible actions, and select the 'best' action for the real robot. To understand how to do this we first need to consider what we mean by the IM's outputs. Clearly, for each action, the robot is likely to have changed its disposition in the WM during the simulation run. Or it may not, either because its move was blocked by another object, or simply because the next action being tested might be 'stand still'. If there are dynamic actors in the environment, then their positions, relative to the robot, are also likely to have changed. Thus the position of the robot at the end of each IM run, and of any other dynamic actors, are useful outputs. Perhaps better still are the changes in position of the robot. But since collisions are significant consequences, as far as safety is concerned, that are likely to be detected directly by the simulator since – during a simulated *what-if* run – those collisions actually happen, then collision or no-collision is another extremely useful output.

Given the outputs of the IM are, minimally, change of position and collision/no-collision, for each next possible action tested, how is the CE to judge which is the best action? Clearly such a judgement requires rules. The rules might, for instance, determine that *all* collisions are unsafe. Thus, for a set of next possible actions tested in the IM if only one has the output no-collision, then that would be chosen and sent to the Robot Controller. But if several next actions are equally safe (i.e. none of them are predicted to result in collisions) how is the CE to decide? The simple answer is that the CE doesn't have to decide between the safe actions, since the RC is, we assume, capable of action selection in order to decide which action is the next best action toward achieving the robot's task or mission. Clearly for n next possible actions modelled and evaluated the number of actions judged safe, s could be any value in the range $(0...n)$, and so the CE needs to send an s-tuple of safe actions to the RC. Using its action selection mechanism, the RC then chooses one of the s-tuple actions (possibly) overriding an unsafe action.

Consider the scenario illustrated in Fig. 1.2. Here the robot is approaching two hazards: a wall on the left and a hole directly ahead. Let us assume the hole is deep enough that it presents a serious hazard to the robot. The robot has four possible next actions, each of which is simulated in its IM. The actions are move ahead left, move straight ahead, move ahead right or remain stationary; for simplicity assume left and right movements actually consist of a turn on the spot, followed by straight moves as shown by the dashed arrows.

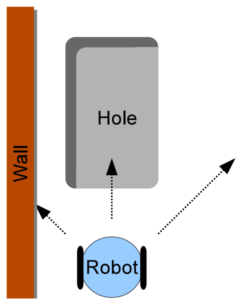

Fig. 1.2 A scenario with static safety hazards

Table 1.1 shows the change of position and collision/no-collision values that might be generated by the IM for each of the four possible next actions of the robot. Two of the four actions are clearly unsafe: *Ahead Left*, which leads to a collision with the wall, and *Ahead*, which results in the robot falling into the hole. It is perfectly reasonable to expect the IM to simulate and detect both outcomes and, from a safety perspective both can be classified as *Collision*; (we can assume the WM's physics engine will model the robot colliding with the bottom of the hole). Two of the actions *Ahead Right* and *Stand Still*, are safe and so the CE will output the 2-tuple (*Ahead Right*; *Stand Still*) to the RC. It is then easy for the RC to select the action *Ahead Right*, since it almost certainly results in the robot moving closer to its target destination.

Robot action	Position Change	Robot Outcome	Interpretation
Ahead Left	5cm	Collision	robot collides with wall
Ahead	10cm	Collision	robot falls into hole
Ahead Right	20cm	No-collision	robot safe
Stand still	0cm	No-collision	robot safe

In the example sketched here some actions are evaluated to be safe (robot outcome: no-collision). What if the situation a robot finds itself in means that all robot actions are evaluated as unsafe, leaving the RC with no options? This problem might be addressed if we arrange that, instead of generating binary (safe or not-safe) outcomes, the CE outputs an analogue value estimating the degree of safety risk. The CE could then provide the RC with the 'least unsafe' options. This approach is outlined in Section 1.4.

1.3.3 *The Object Tracker-Localizer*

The Object Tracker-Localizer (OTL) is required to track the (relative) position of both static and dynamic objects in the robot's local environment, while at the same time localising the robot relative to those objects. Then provide this position data to the IM. For moving (dynamic) objects the OTL must also provide the IM with the speed and direction of those objects, so that their trajectories can be modelled. The OTL is the essential mechanism by which the robot's Internal Model is synchronised to its local environment, and position in that environment. Although the OTL might appear to be a demanding requirement, robots of reasonable sophistication are likely to require sensing and processing of sense data for object tracking and localisation as a part of their normal control architecture.

1.4 Towards an Ethical Robot

Consider the scenario illustrated in Fig. 1.3. Here there are two actors: our self-aware robot and a human. The environment also contains a hole in the ground, of sufficient size and depth that it poses a serious hazard to both the robot and the human. As in the previous example the robot has four possible next actions, each of which is simulated in its IM. Let us extend the architecture of Fig. 1.1 in the following two ways. Firstly, we extend the definition of the 'collision/no-collision' output of the IM, to include *all* safety outcomes, and assign to these a numerical value which represents the estimated degree of danger. Thus 0 indicates 'safe' and (say) 10 'fatal'. An intermediate value, say 4, might be given for a low-speed collision: unsafe but probably low-risk, whereas 'likely to fall into a hole' would merit the highest danger rating of 10. Secondly, we also output, to the CE, the same safety consequence of the other actor(s) in the environment - noting that the way we have specified the IM and its inputs, from the OTL, means that the IM is equally capable of modelling the effect of hazards on *all* dynamic actors in the environment, including itself. If one of those dynamic actors is a human then we now see the possibility of the robot choosing to execute an unsafe action in order to prevent that human from coming to harm.

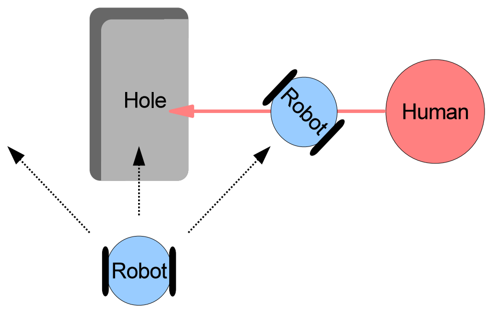

Fig. 1.3 A scenario with both safety and ethical consequences

Table 1.2 shows the safety outcome values that might be generated by

Robot action	Robot outcome	Human outcome	Interpretation
Ahead Left	0	10	robot safe, but human falls into hole
Ahead	10	10	both robot and human fall into hole
Ahead Right	4	4	robot collides with human
Stand still	0	10	robot safe, but human falls into hole

the IM for each of the four possible next actions of the robot, for both the robot and human actors in this scenario. From the robot's perspective, 2 of the 4 actions are safe: *Ahead Left* means the robot avoids the hole, and *Stand Still* means the robot also remains safe. Both of the other actions are unsafe for the robot, but *Ahead* is clearly the most dangerous, as it will result in the robot falling into the hole. For the human, 3 out of 4 of the robot's actions have the same outcome: the human falling into the hole. Only 1 action is safer for the human: if the robot moves *Ahead Right* then it might collide with the human before she falls into the hole.

In order for the CE to generate the action *Ahead Right* in this scenario it clearly needs both a safety rule, as before, and an 'ethical' rule, which can take precedence over the safety rule. This logic might take the form:

```
IF for all robot actions, the human is equally safe
THEN (* default safe actions *)
    output s-tuple of safe actions
ELSE (* ethical action *)
    output s-tuple of action(s) for least unsafe human outcome(s)
```

What we have set out in this section appears to match remarkably well with Asimov's first and third laws of robotics: *(1) A robot may not injure a human being or, through inaction, allow a human being to come to harm, and (3) A robot must protect its own existence as long as such protection does not conflict with the First (or Second) Laws* (Asimov, 1950). The schema proposed here will impel a robot to maintain its own safety (3rd law 'protect its own existence'); it will avoid injuring (i.e. colliding with) a human (1st law 'may not injure a human'), but may also sometimes compromise that rule in order to prevent a human from coming to harm (1st law '...or, through inaction, allow a human to come to harm'). This is not to suggest that a robot which apparently implements part of Asimov's famous laws is ethical in any formal sense (i.e. that an ethicist might accept). But the intriguing possibility of a route toward engineering a minimally ethical robot does appear to be presented.

1.5 Challenges and Open Questions

Although the architecture proposed above is technically realisable with current simulation technology, it is by no means certain that the resulting robot would present itself as a practical proposition. Significant challenges fall into three categories: *performance*, *timing* and *validation*.

Performance Sensor-based simulation is computationally expensive, and the time required to simulate each next possible action and complete the IM cycle is likely to be a major limiting factor on the robot's overall performance. For example, the internal modelling process for the complex anthropomimetic humanoid ECCE-Robot, using workstation grade computational hardware, ran at about one quarter of real-time (Diamond *et al.*, 2012). Ideally we require an efficient internal modelling process that runs in the background, overriding the robot's next control action as and when necessary, yet with no perceptible interruption to the robot's normal operation. Achieving this ideal presents two challenges: engineering the simulation, and integration so that the Internal Model and its data flows integrate smoothly with the robot's actions in the real world.

The key simulation challenge is to find the optimal level of simulation fidelity. Too much fidelity will slow down the simulator; too little and the reality-gap will reduce the value of the simulation outcomes, i.e. the IM will not provide a useful prediction of what will really happen if the robot performs this action. It may be that a variable-fidelity simulator is required, in which the robot can adapt the simulation fidelity according to the perceived hazard. Although current robot simulator technology is likely to be adequate to allow proof-of-principle, a different kind of variable fidelity simulation framework will most probably be needed for a practical real-world robot. Developing this presents a substantial research challenge.

Timing Here we have a number of open questions. Firstly, when and how often does the robot need to initiate the process of internally modelling the sequence of next possible actions? The process is computationally expensive and, given the performance issue discussed above, likely to slow down the robot if it has to wait for the IM cycle to complete before acting. Ideally, the IM cycle would only be triggered as and when some potential hazard is detected by the robot's sensors and, furthermore, far enough in advance that the IM cycle can complete before the robot encounters the hazard. Intuitively, a static environment is safer for the robot, thus we could propose that when the robot senses a nearby object or actor starting to move, the IM cycle is triggered. Such an event would be sensed by the Object Tracker - Localizer, so it would make sense for that process to initiate the whole IM cycle. Perhaps also the OTL should send a signal to the Robot Controller to slow down the robot; an appropriate response perhaps to sensing a moving object but with the benefit of giving longer for the IM cycle to complete.

Secondly, how far ahead should the IM simulate, for each next possible action? Let us call this time t_s. If t_s is too short, the internal modelling process is likely to be useless, since it will not simulate far enough ahead to interact with the hazard that triggered the IM cycle. But setting t_s too long is likely to affect the robot's performance, given the computational cost of Internal Modelling. Ideally t_s and its upper limit (time-out) should be adaptive, but how to set or discover these values clearly presents an open research question.

Validation and verification This chapter is proposing a route to practical self-awareness and, hence, safer robots. But a safety system is worthless unless it can be formally shown to be safe. Thus we face the difficult question of if, and how, a robot engineered along the lines proposed could be validated[2]. At first this might appear to be an insurmountable challenge: after all, the whole idea of the architecture outlined here is that it offers the potential for a robot that can act safely even when it is confronted with new situations, including scenarios not anticipated by the robot's designers. Given that robot behaviours are an emergent property of the interaction between the robot and its environment, then logic would suggest that placing a robot in an unpredictable environment will lead to unpredictable behaviours – and hence a robot that cannot be validated. However, a more careful analysis suggests there may be a route to verification, and hence partial validation.

Firstly, consider that the Generate-and-Test machinery, including its Internal Model, does not control the robot directly. In fact it serves to *reduce* the number of next possible actions in any given situation, by assessing some to be unsafe and inhibiting those in the Robot Controller's action selection mechanism. This suggests that a robot with the Generate-and-Test machinery *cannot be less safe that the same robot without that*

[2]Validation would determine if the robot is safe in use; verification checks the correctness of its design

machinery. If the Robot Controller has already been shown to be safe (or as-safe-as-it-can-be within its design limitations), then the introduction of the internal modelling process cannot compromise that assurance. Let us test this proposition by considering the two possible ways in which the Generate-and-Test process can give incorrect assessments:

- Incorrectly evaluating a safe action as unsafe: here the effect is to (unnecessarily) limit the choice of next possible actions; in effect the robot acts more cautiously than it needs to.
- Incorrectly evaluating an unsafe action as safe: if this action is then selected by the controller, the robot will execute an unsafe action. However, the same robot without the Generate-and-Test machinery would, in the same situation, execute the same unsafe action, so the robot with the Generate-and-Test process is no more unsafe.

How might the internal modelling process might give rise these incorrect assessments above? There are several reasons including at least: (i) the robot might fail to accurately perceive its environment and the objects in it, because of sensor limitations or sensor noise for example, and therefore incorrectly initialise the World Model. (ii) limitations in simulation fidelity (the reality-gap) might result in the Robot Model failing to (virtually) sense an object that has correctly been initialised in the World Model, or failing to (virtually) collide with an object. Or (iii) simulation time t_s is too short, so the IM doesn't simulate far enough ahead to (virtually) encounter a hazard. (i) is clearly a fundamental limitation of any robot; animals and humans also suffer the consequences of sensory limitations. Since perfect perception is impossible (i) cannot be held against the internal modelling approach. (ii) and (iii) are factors discussed above in sections *Performance* and *Timing* and need careful design in order to minimise the likelihood of incorrect assessments.

Secondly, the Generate-and-Test process, including the rule-set in the Consequence Evaluator, are entirely deterministic. Thus for any given situation, i.e. current disposition and set of perceptual inputs, and for a given simulator fidelity and internal timing, a robot will always generate the same set of IM and CE outputs. Thus we may be able to formally check the correctness of the deterministic Generate-and-Test process using agent model checking (Dennis *et al.*, 2012) or deductive verification approaches (Dixon *et al.*, 2002).

1.6 Concluding Discussion

The ideas presented in this chapter have the potential to make progress on three axes: robot safety, self-aware robots and ethical robots.

Robot Safety Designing a robot that can safely interact with humans is a significant and current challenge, which is assuming greater importance with the introduction of workplace assistant robots – robots designed to share a workspace with humans (Alami *et al.*, 2006). Design for safety typically requires an exhaustive analysis of all possible functional hazards. Recent work has extended this approach with a safety system developed during the hazard analysis stage. This safety system, called the safety protection system, is initially used to verify that safety constraints – identified during hazard analysis – have been implemented appropriately. Subsequently the safety protection system serves as a high-level safety enforcer, by governing the actions of the robot and preventing the control layer from performing unsafe operations (Woodman *et al.*, 2012). The internal modelling approach proposed in this chapter circumvents the need for exhaustive hazards analysis: instead the hazards are modelled in real-time, by the robot itself. And since the Internal Model is initialised from the robot's perception of its environment, then, in principle, the robot is able to respond safely to previously unseen hazards in unknown[3] dynamic environments (providing of course the hazards can be perceived by the robot, and the effect of the robot's interactions on those hazards can be tested by the Internal Model). Although this chapter has not explored the potential for learning within the proposed self-aware robot, it is reasonable to extrapolate that the addition of learning mechanisms would not compromise the safety of the robot; indeed they may improve the robot's safety. This contrasts sharply with approaches such as (Woodman *et al.*, 2012), in which learning is a major issue.

Self-aware Robots How self-aware would a robot with the generate and test mechanism proposed in this chapter actually be? The robot would not pass the mirror test, which some argue is a test of self-awareness (Haikonen, 2007). Nor would the robot have any self-reflective self-awareness, or sentience. But this is neither surprising or disappointing. The approach outlined in this chapter is designed only to enable the robot to respond

[3]the robot does of course have prior knowledge: its physics engine means that it can model and predict the outcome of physical interactions

safely to unknown hazards or, with the extension suggested in section 1.4, to behave in some limited sense ethically. Although some theories of consciousness postulate a self-model, for instance (Metzinger, 2009), we can be quite sure that a much richer and more complex set of processes would be required than are present in the architecture outlined here. Nevertheless, we argue that a robot built as proposed in this chapter will be minimally self-aware. Assuming the architecture can be realised as proposed, the robot will – with its Internal Model – be able to test what-if hypotheses about next possible actions, and then moderate its behaviour according to the outcomes of those tests. It is clear from the two example scenarios of Figures 1.2 and 1.3 that without the internal modelling processes, the robot would be unable to choose the safest actions, for itself, or for the human actor in Fig. 1.3. The robot is – we contend – minimally but *sufficiently* aware of itself and its immediate environment, and the consequences of its actions, to merit the label self-aware. Thus, although aimed at safer robots, the ideas of this chapter do have the potential to advance work in self-aware robots; perhaps most of all by exploring the difficult simulation and timing challenges outlined in section 1.5.

Ethical Robots In introducing their seminal book Moral Machines, Wallach and Allen write (2009): "A concern for safety and societal benefits has always been at the forefront of engineering. But todays systems are approaching a level of complexity that, we argue, requires the systems themselves to make moral decisions. ... This will expand the circle of moral agents beyond humans to artificially intelligent systems, which we will call Artificial Moral Agents (AMAs)."

Wallach and Allen go on to outline the key engineering challenge (italics added): "... wherever one comes down on the question of whether a machine can be genuinely ethical (or even genuinely autonomous), an engineering challenge remains: *how to get artificial agents to act as if they are moral agents*." and then to express this engineering challenge in terms of action selection: "If multipurpose machines are to be trusted, operating untethered from their designers or owners and programmed to respond flexibly in real or virtual world environments, there must be confidence that their behaviour satisfies appropriate norms. This goes beyond traditional product safety ... *if an autonomous system is to minimise harm, it must also be 'cognisant' of possible harmful consequences of its actions, and it must select its actions in the light of this 'knowledge'*, even if such terms are only metaphorically applied to machines." The approach set out in this chapter may provide initial steps toward the engineering realization of Wallach and Allen's Artificial Moral Agent. An ethical robot might not simply be science fiction after all.

In summary, this chapter has argued that the design of safer robots for unpredictable (i.e. human) environments requires mechanisms for self-awareness. One such mechanism, the internal model – a self-simulation in which both the robot and its environment are continuously modelled – allows the robot to model and hence evaluate the possible consequences of its next actions. The chapter has proposed an architecture in which the internal model does not control the robot directly, but instead inhibits those next actions it assesses to be unsafe – thus reducing the number of actions available to the robot controller's action selection mechanism. Given that robot simulation technology is already reasonably advanced, the proposed architecture is offered as a practical proposition, although not without significant implementation challenges. The chapter has also proposed a surprisingly simple extension that would, in principle, allow a robot to prevent physical harm coming to a human in its vicinity – such a robot would thus be safe and, at least minimally, ethical. Difficult questions such as how self-aware (or how ethical) such a robot would really be, and how that self-awareness would be tested, are left to future work. Another difficult open question considered in the chapter is that of validation: could a robot built along the lines proposed here be proven to be safe? While formal verification of elements of the robot's Internal Model may be possible, full validation of the robot as a whole might not. We should however, be confident that the mechanisms for self-awareness outlined here would lead to a robot that is demonstrably safer than a robot without such mechanisms.

Bibliography

Alami, R., Albu-Schaeffer, A., Bicchi, A., Bischoff, R., Chatila, R., Luca, A. D., Santis, A. D., Giralt, G., Hirzinger, G., Lippiello, V., Mattone, R., Sen, S., Siciliano, B., Tonietti, G. and Villani, L. (2006). Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges, in *Workshop on Physical Human-Robot Interaction in Anthropic Domains, at the Int. Conf. on Intelligent Robots and Systems (IROS 2006).*

Asimov, I. (1950). *I, ROBOT* (Gnome Press).

Bongard, J., Zykov, V. and Lipson, H. (2006). Resilient machines through continuous self-modeling, *Science* **314**, 5802, pp. 1118–1121.

Dennett, D. (1995). *Darwin's Dangerous Idea* (London: Penguin).

Dennis, L. A., Fisher, M., Webster, M. and Bordini, R. H. (2012). Model checking agent programming languages, *Automated Software Engineering* **19**, 1, pp. 5–63.

Diamond, A., Knight, R., Devereux, D. and Holland, O. (2012). Anthropomimetic robots: Concept, construction and modelling, *International Journal of Advanced Robotic Systems* **9**, pp. 1–14.

Dixon, C., Fisher, M. and Bolotov., A. (2002). Resolution in a logic of rational agency, *Artificial Intelligence* **139**, 1, pp. 47–89.

Haikonen, P. O. (2007). Reflections of consciousness: The mirror test, in *Proceedings of the 2007 AAAI Fall Symposium on Consciousness and Artificial Intelligence*, pp. 67–71.

Holland, J. (1992). *Complex Adaptive Systems* (Daedalus).

Holland, O. (ed.) (2003). *Machine Consciousness* (Imprint Academic).

Isidori, A., Marconi, L. and Serrani, A. (2003). Fundamentals of internal-model-based control theory, in *Robust Autonomous Guidance*, Advances in Industrial Control (Springer London), pp. 1–58.

Jacobi, N., Husbands, P. and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics, in *Proceedings of the Third European Conference on Advances in Artificial Life,* (Springer), pp. 704–720.

Marques, H. and Holland, O. (2009). Architectures for functional imagination,, *Neurocomputing* **72**, 4-6, pp. 743–759.

Metzinger, T. (2009). *The Ego Tunnel* (Basic Books).

Michel, O. (2004). Webots: Professional mobile robot simulation, *International Journal of Advanced Robotic Systems* **1**, 1, pp. 39–42.

O'Dowd, P. J., Winfield, A. F. T. and Studley, M. (2011). The distributed co-evolution of an embodied simulator and controller for swarm robot behaviours, in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, pp. 4995–5000.

Stepney, S., Welch, P. and Andrews, P. (eds.) (2011). *CoSMoS 2011: Proceedings of the 2011 Workshop on Complex Systems Modelling and Simulation,* (Luniver Press).

Vaughan, R. T. and Gerkey, B. P. (2007). Really reused robot code from the player/stage project, in D. Brugali (ed.), *Software Engineering for Experimental Robotics* (Springer), pp. 267–289.

Vaughan, R. T. and Zuluaga, M. (2006). Use your illusion: Sensorimotor self-simulation allows complex agents to plan with incomplete self-knowledge, in *Proceedings of the International Conference on Simulation of Adaptive Behaviour (SAB)*, pp. 298–309.

Wallach, W. and Allen, C. (2009). *Moral Machines: Teaching Robots Right from Wrong* (Oxford: Oxford University Press).

Woodman, R., Winfield, A. F. T., Harper, C. and Fraser, M. (2012). Building safer robots: Safety driven control, *International Journal of Robotics Research* **31**, 13, pp. 1603–1626.

Zagal, J. C., Delpiano, J. and Ruiz-del Solar, J. (2009). Self-modeling in humanoid soccer robots, *Robot. Auton. Syst.* **57**, 8, pp. 819–827.

Index