Stakeholder Groups in Computational Creativity Research and Practice

Simon Colton[1], Alison Pease[2], Joseph Corneli[1],
Michael Cook[1], Rose Hepworth[1] and Dan Ventura[3]

[1] Computational Creativity Group, Department of Computing
Goldsmiths, University of London, UK
ccg.doc.gold.ac.uk

[2] School of Computing
University of Dundee, UK

[3] Computer Science Department
Brigham Young University, USA

Abstract. The notion that software could be independently and usefully creative is becoming more commonplace in scientific, cultural, business and public circles. It is not fanciful to imagine creative software embedded in society in the short to medium term, acting as collaborators and autonomous creative agents for much societal benefit. Technologically, there is still some way to go to enable Artificial Intelligence methods to create artefacts and ideas of value, and to get software to do so in interesting and engaging ways. There are also a number of sociological hurdles to overcome in getting society to accept software as being truly creative, and we concentrate on those here. We discuss the various communities that can be considered stakeholders in the perception of computers being creative or not. In particular, we look in detail at three sets of stakeholders, namely the general public, Computational Creativity researchers and fellow creatives. We put forward various philosophical points which we argue will shape the way in which society accepts creative software. We make various claims along the way about how people perceive software as being creative or not, which we believe should be addressed with scientific experimentation, and we call on the Computational Creativity research community to do just that.

1 Introduction

It seems uncontroversial to state that one of the long-term goals of research into Computational Creativity is to see creative software embedded in society: Apple's iTunes will one day compose new music for us, rather than just recommending it; Microsoft's PowerPoint will suggest jokes for a speech we're writing; videogames will be constructed on the fly to fit our preferences and mood; software will routinely make scientific discoveries; and household appliances will be endowed with creative abilities, like a refrigerator able to concoct a recipe to fit its contents. It is also uncontroversial to point out that another long-term goal of the field is to further our understanding of human creativity, both individually and in societies, through computer simulation.

Computational Creativity researchers have made steady progress towards software which creates, by employing, advancing and inventing novel Artificial Intelligence, natural language processing, graphics, audio and other techniques for creative purposes. There is, of course, much progress still to be made technically, so that software can be creative and be seen to be creative, in order for consumers to be provided with valuable artefacts and enjoyable creative experiences. In addition to the technological hurdles faced, it is clear that certain sociological issues stand in the way of progress. That is, people naturally tend towards thinking that nuts-and-bolts, bits-and-bytes machines will never have a creative spark, and different sets of people instantiate this tendency in different ways. Through much engagement and outreach, we have come to the conclusion that understanding people's conceptions of software being creative is an important tool to be used towards the long-term goal of understanding human creativity, and that favourably guiding these conceptions will be essential in bringing about the long-term goal of embedding creative software in society.

In largely separate tracks of research, we have examined how creative software is perceived by three different types of *creativity stakeholders* – people who may have something to gain or lose from software which is creative – from a practical and a philosophical perspective. We address the different types of creativity stakeholders in general in section 3, and concentrate in the rest of the chapter on three particular types. In particular, in section 4, we address members of the general public exposed to creative software. Following this, in section 5, we address observer issues within Computational Creativity research itself. Finally, in section 6 we address videogame designers, as an exemplar of a focused community of creative individuals within which creative software has begun to make an impact. We posit that, because of the different issues that each stakeholder community raises with creative software, it currently helps to study them independently, and suggest approaches to altering the perception that people have of software in these groups in different ways. However, by bringing together these strands for the first time here, we can begin to discuss more unified approaches to the presentation of software written to be autonomously creative.

Throughout this chapter, we propose hypotheses about how each set of stakeholders perceive software as being creative or not, based on practical experiences, philosophical studies and theoretical advances. We believe that our arguments in favour of these claims are sufficiently strong for them to be taken to the next level and tested scientifically through observer-based experimentation – and that the hypotheses provide an agenda the Computational Creativity research community cannot ignore. To conclude in section 7, we suggest some practical ways in which these claims (which are presented as numbered hypotheses) could be investigated. In order to explain and support the claims we make, in the next section, we first present a philosophical perspective on the notion of creativity, which will introduce ideas that underpin the material in the rest of the chapter.

2 A Perspective on Creativity

We hold that creativity is a *secondary* and *essentially contested* quality of a person, and that linguistic usage of terms related to creativity can often be declarative illocutionary speech acts. We unpack these assertions below. Firstly, we believe that attributions of creativity are contextualist, having no truth value which is independent of context, perception and interpretation. In this way we see creativity attributions as analogous to the Lockean notion of a secondary quality [40]. Locke distinguished *primary* and *secondary qualities*, where the former are taken to be intrinsic to an object, for example, its mass, and the latter are understood to be perception-dependent, for example, colour. While these Lockean qualities are directly tied to sensory perception, as opposed to the aesthetic and social category of creativity, the distinction is still a useful one here, since it highlights different types of properties. Dennett's intentional stance [23] is also of interest here: we may adopt a "creativity stance" towards a person and interpret their work *as though they were being creative*, in order to better understand (rather than predict) their behaviour. Likewise, we may find that the "creativity stance" provides a new way of understanding the behaviour of a piece of software which goes beyond the physical details of the program.

Gallie introduced *essentially contested concepts* as those for which "the proper use . . . inevitably involves endless disputes about their proper uses on the part of their users" [29, pp. 169], to which Gray added that the disputes ". . . cannot be settled by appeal to empirical evidence, linguistic usage, or the canons of logic alone" [30, pp. 344], and Smith noted that ". . . all argue that the concept is being *used inappropriately* by others" [56, pp. 332]. In the *Cambridge Handbook of Creativity*, Plucker and Mabel assert that:

> . . . despite the abundance of definitions for creativity and related terms, few are widely used and many researchers simply avoid defining the relevant terms at all. [49, p 48]

Clearly, certain notions such as *art* are essentially contested concepts, looking at the multitude of articles written each year in the popular and cultural press asking: "But is it Art?" Indeed, Gallie points out that the statement: "This picture is painted in oils" can be disputed whilst the disputants nevertheless agree on the proper usage of the terms involved, whereas the assertion "This picture is a work of art" is likely to be contested

> . . . because of an evident disagreement as to – and the consequent need for philosophical elucidation – of the proper general use of the term "work of art" [29, pp. 167].

As a recent example, the question of whether videogames should be classed as art was raised by a Guardian art critic [33], to which the Guardian games editor responded:

> Here is a good way to tell if a critic is having a moment of madness: they will attempt to define art. The greatest philosophers in history have floundered on the question, many simply avoided it altogether, preferring to grapple with more straightforward questions – like . . . the existence of God. Art is ethereal, boundless, its meaning as transient as the seasons. When you think you have grasped it, it slips through your fingers [58].

While this is only one example, it serves as an exemplar of the kinds of debates that occur daily about the nature of art. While it is true that the preoccupation with expressing creativity is a relatively modern aspect of the visual arts, if the notion of *art* is indeed essentially contested within our culture, then the notion of the *creativity* that went into producing a given artwork should be seen accordingly. In particular, a selection of criteria for what counts as creativity is required in any coherent scheme for understanding and evaluating creativity in art. This is the perspective advanced by Jordanous [34], with which we agree – although we also agree with her point that there is unlikely to be broad and lasting agreement about just what the precise criteria of creativity actually are. We can further justify the idea that proper usage of the term *creativity* involves endless debate about its proper usage by reference to the multitude of volumes written about improving, managing and assessing creativity in people, organisations and society. Indeed, as a society, we are better off if we do not agree about what creativity means – in the sense that the disputes we have about this are an engine for change and progress, and it would surely be stultifying if we all suddenly agreed on this most important of concepts. While it is problematic for various areas of study – not least Computational Creativity – that creativity is an essentially contested quality of any person, it is something we need to embrace and even celebrate. For more in-depth discussion of these issues, see Jordanous [34, chapter 3]. We may ask, in practice, what does it mean to *say* someone or something is "creative"? Austin informally introduced the notion of an *illocutionary act* as a locution that also serves to perform another action [4]. Searle further categorised such speech acts into: *assertives*, *directives*, *commissives*, *expressives* and *declarations* [54]. *Declarations* in particular are understood to change reality in accordance with the proposition stated. An example of such a speech act is: "I pronounce you husband and wife." We believe that – in certain circumstances – people can bestow the reality of a person being creative simply by stating it. To see this, we recall the contested nature of creativity, and the assumption that there is no general consensus about what makes someone creative. It follows that people who are not particularly invested in the creativity (or lack thereof) of someone else may be swayed by the declarative speech act of a third party in a position of authority. When Nicholas Serota, long time director of the Tate art museums and galleries, says that a piece is a great work of art, that work becomes (at least temporarily) a great work. When he states that a particular artist is unusually creative, who are we to argue? Given that the sentence 'X is creative' is often shorthand for: 'Most people agree that they perceive X to be creative', such authorities can essentially bring into being the creativity of X, regardless of whether X perceives him/herself as creative or not.

3 Communities of Creativity Stakeholders

In order to understand the different groups of creativity stakeholders, the relationships between them, and the ways in which meaning is continually being created, negotiated and re-created, we can look to sociology. In Latour's *Actor Network Theory* [37], he describes such stakeholders and diverse social groups as actors in a network. Meaning is created socially via actors who cluster into diverse stakeholder groups. These groups are in constant flux, as relationships, actors and ideas within the groups change and come into conflict with each other. Latour holds that understanding such dynamics in the network is essential to understanding processes of innovation and knowledge-creation in science and technology. The process by which a network is formed and comes to be represented as a single entity is called *translation*, and is a key concept in the Actor Network Theory. Translation consists of various phases: the initial formation of a programme and identification of actors in a new network with a novel, shared goal (*problematisation*); the strengthening of the network via formal and informal means (*interessement*); ways of evolving the network and providing structures for new members to join (*enrolment*); and acquiring the resources and power to build an effective institution which can achieve its goal (*mobilisation*).

In the case of Computational Creativity, relevant creativity stakeholders include researchers, the wider AI community, funding bodies, experts in the psychology of human creativity, neuroscientists, artists, art critics, journalists, philosophers, educators, the public, and so on. Each group has accompanying visions, beliefs and goals, in which they have, to a varying degree, invested (and which, to a varying degree, define them as a group). We hold that understanding such different perspectives and their interactions is essential if software is ever to be deemed creative by mainstream consumers of cultural artefacts. In this section, we consider these stakeholder groups and in particular use Latour's notion of translation to look at how Computational Creativity researchers have evolved into a community. We also look at some of the relationships between the groups, both in the context of Computational Creativity and the wider scientific arena.

3.1 The Computational Creativity Stakeholders

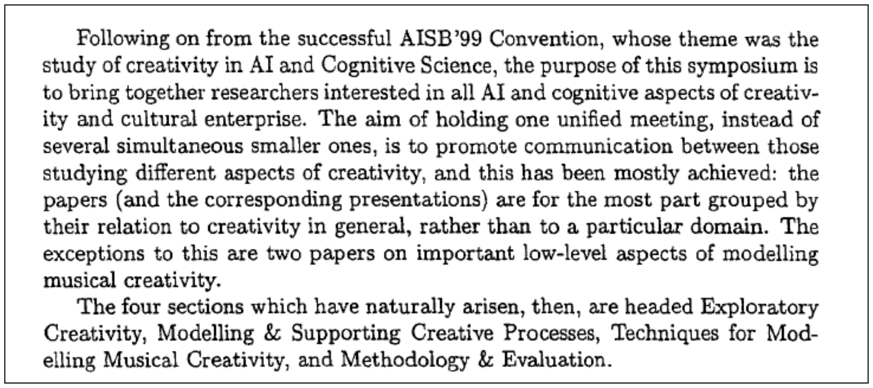
Members of the Computational Creativity community are largely people with a background in Artificial Intelligence or computer science and an interest in creativity. They are usually professional academics with the infrastructure of a university supporting them. AI is itself a young field – originating in the 1950s – and, since initial attempts to build general intelligence machines, has fragmented into many different specialisations and subdisciplines: once established, these then form the *internal environment* for any new area, in terms of providing ideas, methods and concepts, and at times, competition. Academic measures of the health of such subdisciplines include the amount of funding awarded, the number of lectureships or professorships in the field, the existence of a journal and an international conference series, and other scientifically respectable incentivisation schemes and recognition.

It was against this backdrop that AI researchers with an interest in creativity found themselves in the late 1990's. Given their background, they were not only accustomed to the idea that machines can be intelligent, but their very livelihood depended on that premise. So it was not, perhaps, such a huge leap to the idea that machines can be creative. However, since there was no infrastructure supporting research into Computational Creativity, early researchers largely had to establish their reputation in different (possibly related) areas of AI and build up the Computational Creativity community almost on their own time, sometimes taking considerable career risks to do so.

Latour's notion of *translation* can help us to understand how the community formed. *Problematisation* occurred when a few core people identified the goal of building creative software as a subdiscipline of AI. Between them, they had the influence and organisational power to make *Creativity in AI and Cognitive Science* the theme of the AISB'99 Convention (co-chaired by Geraint Wiggins, Helen Pain and Andrew Patrizio). This featured a keynote address by Margaret Boden, a cognitive scientist known for her popular writing on creativity in people and in machines [5,6]. The initial symposium was followed up by four further events[1] held at AISB'00 - AISB'03, and a series of workshops on creative systems at major AI conferences. We present an extract from the editors' introduction to the Proceedings of the Symposium on Creative and Cultural Aspects of AI and Cognitive Science, held at AISB in 2000 in figure 1 below. This was the *interessement* phase. These were further consolidated with the International Joint Workshops on Computational Creativity (IJWCC), held 2004-08, during which time the community grew from twenty, or so, to double that (*enrolment*). Finally, the community was considered healthy enough, strong enough and large enough to launch the first International Conference on Computational Creativity in 2010. For a history of the field up to this stage, see [8] in a special issue of the AI magazine on Computational Creativity.

The community continues to evolve and grow, with the series having recently held its Fifth Annual International Conference (2014), with around 90 delegates. In order to organise and guide the international series, a Steering Committee was set up consisting of anyone who had chaired an IJWCC event, and they formed the Association for Computational Creativity (ACC) in 2010 and set out rules which enabled new members to join and old members to leave the Association (*mobilisation*). Landmark events during this time included the first ever award of a Chair in Computational Creativity (to Geraint Wiggins, in 2004, by Goldsmiths, University of London) (only one of two - the other being held by Simon Colton also at Goldsmiths, University of London, awarded in 2013); the first PhD with Computational Creativity in its title (Anna Jordanous, University of Sussex, 2012 [34]) and the first NSF and EU calls for proposals in Computational Creativity (*CreativeIT, NSF Program Solicitation 09-572* [1] and *Objective ICT-*

[1] *Creative and Cultural Aspects of AI and Cognitive Science* (2000) and then simply *AI and Creativity in Arts and Science* (2001 - 2003)


Following on from the successful AISB'99 Convention, whose theme was the study of creativity in AI and Cognitive Science, the purpose of this symposium is to bring together researchers interested in all AI and cognitive aspects of creativity and cultural enterprise. The aim of holding one unified meeting, instead of several simultaneous smaller ones, is to promote communication between those studying different aspects of creativity, and this has been mostly achieved: the papers (and the corresponding presentations) are for the most part grouped by their relation to creativity in general, rather than to a particular domain. The exceptions to this are two papers on important low-level aspects of modelling musical creativity.

The four sections which have naturally arisen, then, are headed Exploratory Creativity, Modelling & Supporting Creative Processes, Techniques for Modelling Musical Creativity, and Methodology & Evaluation.

Fig. 1. An excerpt from the preface of the Proceedings of the Symposium on Creative and Cultural Aspects of AI and Cognitive Science, held at AISB in 2000, written by Geraint Wiggins. Note the 'natural' emergence of themes within the field, although of course these are very much subject to the Call for Papers, the communities who received the call, the instructions given to the reviewers, the reviewers themselves and the editor's vision.

2013.8.1, Technologies and scientific foundations in the field of creativity [2, p. 81]). The process has been carefully managed throughout, with an eye on political as well as intellectual developments. Social factors have also played a key role, being inextricably linked to internal development of scientific knowledge [38].

3.2 Other Creativity Stakeholders

Each of the other stakeholder groups will have a similarly fascinating history. Some, such as the EU funding body, are tightly bound and have a formal definition of themselves and their goals. Others, such as the general public – for whom the concept of *translation* is meaningless – are much more loosely defined. Of note is who the decision makers are in each of these groups. In the Computational Creativity community, it is clear that a few people have had a huge influence, and it is likely that this is also the case for other groups of stakeholders. It may be worth considering these in detail, especially from a point of view of motivation and power. For instance, Boden's way of seeing creativity dominated the first decade of the community growth. Likewise, a few core individuals working for the EU had the influence to prioritise research into Computational Creativity, and to fund around €10m worth of projects.

• The general public

When describing what they do, to a layman, most researchers into Computational Creativity will probably have experienced reactions such as: "A computer that is creative might be dangerous – it might kill us"; "Creativity is a celebration of humanity, and the very idea of Computational Creativity cheapens that";

"I read a poem or listen to music to communicate with another human being. I don't want to communicate with a computer, I want a live human connection", and so on. It is important to determine where these ideas come from, whether they are grounded in anything, whether we should try to counter them, and if so, how? While such emotional responses are not necessarily negative, it might be the case that they hinder reasoned debate. Public perception of Computational Creativity derives from multiple sources, including journalistic coverage (or lack of it), science fiction narratives, opportunities to consume computationally created artefacts and so on. We look further at observer issues in the general public in section 4 below.

- **Fellow creatives**

Creative people sometimes voice the worry that "Computers are going to put us out of a job". This group is similar to the general public in terms of influences and attitudes. It seems that artists might be being encouraged to worry about software replacing them, because such sensationalist stories sell newspapers. We study a particular community of creative people, namely videogame designers in section 6 below.

3.3 Relationships Between the Different Stakeholder Groups

There have been several interactions between the Computational Creativity community and members of the public and fellow creatives. For instance, Colton and Ventura hosted a festival of Computational Creativity in 2013, *You Can't Know my Mind* [10], and other events have followed on from this. Historical relationships between scientists and the public can also elucidate current interactions. In other fields, there have been some explicit campaigns to manufacture doubt, by parties who are threatened by specific scientific advances. For instance, the tobacco industry tried to discredit and discourage the notion that smoking is bad for our health; likewise the fossil fuel industry did the same in the case of global warming. Here we see that a few powerful actors can sometimes bring an entire body of established scientific knowledge into question.

Ravetz argues that scientific ignorance may in some ways be as prone to social construction as scientific knowledge [50,51], cited in [57, p. 37]. Stocking and Holstein [57] explore different perceptions that journalists have of their roles, concluding tentatively that journalists construct scientific ignorance consistent with their own interests. Even without such dark agendas, there are other examples from the history of science in which public perceptions conflict with scientific thinking and have been managed, or controlled, in order to bring them into line with current scientific results. Famous examples in which scientific advances have challenged our image of ourselves and our universe include Copernicus's heliocentric model, which challenged our view that the earth is the centre of the universe; Darwin's theory of evolution, which challenged concepts of what it means to be human, to be distinct from other animals, and the notion that our existence has a higher purpose; and Lemaître *et al.*'s Big Bang theory, which challenged the view that the universe is a stable, stationary entity. In all of these cases the scientists faced their own challenges of reconciling their findings with their religious or world views, and then a process of outreach was necessary in order to gain wider social acceptance. Thus, we see Thomas Huxley – "Darwin's bulldog" – promoting Darwin's theory in the face of many varied and negative responses to it (some of which are recorded in [25]) and helping it to gain wider acceptance, transitioning from scientific to social fact. Today, people in the fields of genetically modified food and stem cell research endeavour to gain wider social acceptance in the form of media coverage and well-funded outreach programmes aimed at educating both school children and the wider community.

Computational Creativity is in a particularly difficult position, since its main research question concerns an essentially contested concept. On certain understandings, the question "can machines be creative?" may be answered negatively, without further elaboration or debate. Thus, we see part of the job of the Computational Creativity community consisting in the delivery of outreach programmes, in which creative software is demonstrated and explained, and the artefacts it has produced exhibited in a setting in which consumers of creative artefacts might begin to appreciate them. In [28], Franzen *et al.* explore the impact that such dissemination activities can have on scientific progress, and argue that the right name, image or metaphor has the power to make or break relations between a scientific discipline and the public. For instance, consider Dolly the sheep from the Roslin Institute in Edinburgh and Ida the primate fossil from the Messel Pit in Germany. These names make it easier for the discoveries to be visualised and discussed. Arbib and Hesse go further, stating that "scientific revolutions are, in fact, metaphoric revolutions" [3, p. 156], cited in [27, p. 5].

In addition, then, to sociological narratives, it is important to consider language use by each stakeholder group. The role of spin doctors is well-known in the political arena, in which those who bestow power are influenced in their thinking by vocabulary, metaphors and frames. In our case, the public have the power to bestow or withhold the word "creative" when describing software. Thus, we need to consider the language that we use. Lakoff [35] argues that we fit new information into pre-existing frames, which are built up slowly over time, and if we don't have appropriate frames, then we might misunderstand the information. Using the wrong frame, which is triggered by specific vocabulary, even to deny a message, only reinforces the frame. Thus, rather than trying to argue that "creative software is not scary", we should build up our own vocabulary, frames and metaphors for thinking about it.

Hypothesis 1 *Different stakeholder groups (including Computational Creativity researchers, the general public, domain creatives, psychologists, philosophers, educators, critics, journalists, bureaucrats, etc.) assess creativity in software differently, and there is no one-size-fits-all approach to presenting what software does and what it produces in the best way to increase perception of creativity.*

Given this, we believe it is currently appropriate to study stakeholder groups separately, as we do in the following sections.

4 Observer Issues with the General Public

We introduce here three notions, namely *essential behaviours*, *the humanity gap* and *software accounting for its actions*. We believe these are important in understanding how people generally react to the idea of software being creative, and thus are important in managing and shaping those reactions. To end the section, we present a case study in handling public perception of creativity in software, and we introduce another notion, namely that of *accountable unpredictability*.

A working definition of the field of Computational Creativity research as a subfield of Artificial Intelligence research given in [21] is as follows:

> The philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative.

While this definition is not universally accepted (with a challenge to focus on system-level creativity rather than individual responsibilities given in [32]), variations of it have been used to describe the field for many years.

The usage of the word ‘unbiased’ in the above definition hints at a problem encountered in evaluating projects where generative software produces *artefacts* (poems, paintings, sonatas, recipes, theorems, etc.) for human consumption. In particular, people generally have natural biases against, but also occasionally in favour of, artefacts produced by computers over those produced by people. In particular, negative, so called ‘silicon’, biases have been observed under experimental conditions [24,43]. Hence, in stipulating that observers must be unbiased, the definition above emphasises a scientific approach to evaluating progress in the building of creative systems, whereby experimental conditions are imposed to rule out, or otherwise cater for, such biases. One such experimental setup is the *Turing-style comparison test*, where computer-generated and human-produced artefacts are mixed and audience members make choices between them with zero context given about the processes involved in their production. It is seen as a milestone moment if audiences cannot tell the difference between the artefacts produced by people and those produced by a computer. We believe there are many problems in the application of such tests in the general context of presenting the processing and products of creative software, as expanded in the subsections below.

4.1 Essential Behaviours

We suggest not asking people if they believe software is behaving creatively, but rather concentrating on whether they perceive the software to be acting *uncreatively*. Using our standpoint above that the notion of creativity is essentially contested [29], we expect that no matter how sophisticated our software gets, we will not see consensus on such matters. However, we have found that people agree much more on notions of uncreativity: if a program doesn’t exhibit certain behaviours onto which certain words can be projected, then it is easy to condemn it as being uncreative. Building on the foundational arguments given in [14], we propose that audience members can too easily label software as uncreative if they are unable to project any of the following words onto the behaviours they perceive software to be exhibiting:

skill, *appreciation*, *imagination*, *learning*, *intentionality*,
accountability, *innovation*, *subjectivity* and *reflection*

We have found that assessing the level of projection of these words onto the behaviours of software can help us to gauge people's opinions about (the lack of) important higher-level aspects of software behaviour, such as autonomy, adaptability and self-awareness. Note that we make no claim about the above behaviours being sufficient for a perception of creativity: a necessary set of behaviour types for avoiding the uncreativity label is not the same as a sufficient set of behaviour types for gaining the creativity label. This mis-interpretation of our aims for highlighting the above essential behaviours has propagated somewhat, for instance in [7].

Hypothesis 2 *Creativity in people and software is essentially contested and secondary, and hence it might be advantageous to work on people's perception of* uncreativity *in software, as this is easier to predict/manage. Software exhibiting the essential behaviour types highlighted above is necessary for it to avoid being labelled as uncreative. Eventually, when there are no good reasons to label software as uncreative, people may choose to label it as creative.*

4.2 The Humanity Gap

One could argue that, given the particularly human-centric nature of creativity, and that a human connection is paramount in much of the arts, it is simply inappropriate to use the term 'creative' to describe software. The status quo is that we currently haphazardly apply human terminology related to creativity to software, which often requires the projection of other human qualities onto software, such as it being juvenile, which is inherently error prone, given that computers are patently not people. Another option is to ignore the non-human nature of software and concentrate on what it produces, rather than on what it is, or what it does. To begin to address the kind of silicon biases described above, researchers often compare the interpretation of computer-generated and human-produced artefacts in a rather extreme "blind experiment" situation in which knowledge about the personality of the artist and their practice is entirely missing. The philosophical grounding of such an approach [59,60] matches the motivation of several art movements [26,36] and many individual artists who have expressed a desire for their work to be taken at face value (see [17] for examples and further discussion).

We argue that in modern culture, a curious thing can happen when artists attempt to remove all reference to themselves and their process from discussions about the artistic (and commercial) value of their work. That is, in the absence of such information, people may tend to fill in the gaps about personality and process, and may do so in ways which bolster the credibility of an artist and increase the perceived value of his/her works. Indeed, one could argue that – in the same way that artists invite people to interpret the imagery in artworks in their own way by not prescribing what people should see/read/hear, in refusing to provide meta-level details about personality and process, artists, writers and musicians are actually (purposefully or not) inviting art lovers to invent interesting and engaging back-stories about who they are and what they do.

In such a context of non-disclosure, the comparison of the situation for computer-generated artefacts with the situation for human-produced artefacts is not particularly favourable. The vast majority of people have little or no idea about programming or programs, and may even harbour a desire not to find out about these things. Thus, when invited to assess a computer generated painting or poem, say, without background knowledge, they are denied any opportunity to invent a back-story, as they cannot project personality traits or romantic situations onto the computer, and cannot enter into any dialogues. More importantly, this situation can lead to people realising how much they value the human connection, whether actual or imagined, in such situations. We posit that there is a *humanity gap* that must be faced by Computational Creativity researchers who want their software to enhance society by being creative for artistic and utilitarian purposes.

Turing-style experiments, which epitomise the practice of non-disclosure, are intended to reduce variables so that a scientific study of the value of computer generated artefacts can be undertaken. One could argue that these contexts are intended to help people realise how much they value the aesthetic appeal of art, literature and music, regardless of other factors. This may be true, but we believe that such tests can actually help people realise how little they can relate to the computational origin of artefacts. In [46], we raise other issues with Turing-style comparison studies: in particular, we suggest that they encourage naïvety and pastiche generation in creative software. As a final point, it is clear that such experimental conditions are not sustainable if we are to enhance society with creative software. In the long term, biases about machine creation need to be embraced and managed, rather than factored out through experimental setups.

Hypothesis 3 *Turing-style comparison tests serve to highlight the humanity gap, and while they might serve short-term scientific gain, they are damaging to the long-term goal of embedding creative software in society.*

4.3 Software Accounting for its Actions

We argue in [9,14] that people take into account how a person or software operates when they assess the value of the output it produces. To address this issue, we advocate a development path to follow when building creative software: (i) the software is given the ability to provide additional, meta-level, information about its process and output, e.g., giving a painting or poem a title (ii) the software is given the ability to write commentaries about its process and its products

(iii) the software is given the ability to write stories – which may involve fictions – about its processes and products, and (iv) the software is given the ability to engage in dialogues with people about what it has produced, how and why. Indeed, giving software the ability to discuss its creative works would mirror Turing's original proposal for an intelligence test [59] to a greater extent than tests focusing only on consumer perception of artefacts. As a preliminary example, in [18], we demonstrated a poetry generation system which is able to provide commentaries about its poetry, and how and why it produced a particular poem.

As we discuss in [47], in a computational setting, there are advantages to software being immersed in environments where serendipity might occur. However, accounting for lucky events that trigger creative acts may actually lessen the celebration and hence the impact that the acts have. It is important to note that people tend not to describe their processes and products in the explicit way we advocate for software, preferring to maintain some level of mystery. Nevertheless, we believe that, at this stage in the development of computationally creative systems, it is important to address the humanity gap – without aspiring to eliminate it. *Framing* [9] serves to highlight that intelligent processing was used to produce artefacts, which is an important first step. Given that audience members will typically not be able to come up with an interesting backstory without some scaffolding, positive acts of framing are likely to have more fruitful impact than an overall air of mystery.

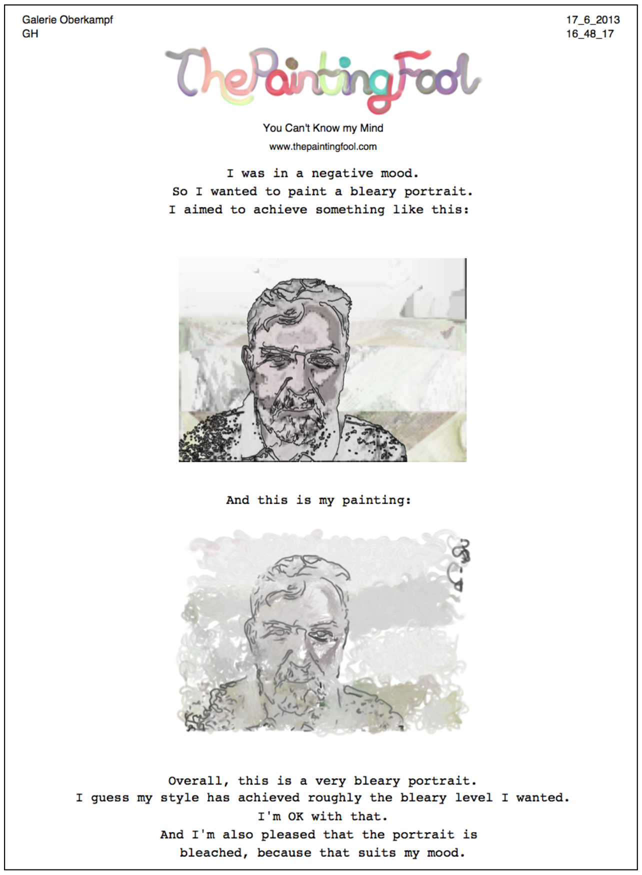
Another possible way to address the humanity gap is to manage people's expectations about the level of humanity they will encounter through a computationally produced artefact. In the same way that when people buy an *e-book* they know they are not going to get a physical object, we advocate telling audiences that they are reading a *c-poem*, and hence – in the knowledge that it was produced computationally – they will get a reduced human connection. We can go further in *re-imagining* traditional artefacts, for instance in suggesting that a c-poem is actually a doublet of texts, one which resembles a traditional poem and another which provides a commentary about the motivations, actions and results of the software's processing. We believe this will highlight the humanity gap, but that it will do so in such a way as to help people to engage with and appreciate the creative process, and better enjoy the artefacts produced by software.

Hypothesis 4 *The humanity gap can be addressed by re-imagining the nature of creative artefacts, to manage expectations of humanity. In particular, it is advantageous for software to account for its processes and products through additional material such as a commentary.*

4.4 A Case study in Automated Portraiture

As part of an exhibition with The Painting Fool[2] system [16] in 2013, we enabled the software to produce portraits for people live in a gallery, as described in [10].

[2] Online presence: `www.thepaintingfool.com`



Galerie Oberkampf
GH

17_6_2013
16_48_17

ThePaintingFool

You Can't Know my Mind
www.thepaintingfool.com

I was in a negative mood.
So I wanted to paint a bleary portrait.
I aimed to achieve something like this:

And this is my painting:

Overall, this is a very bleary portrait.
I guess my style has achieved roughly the bleary level I wanted.
I'm OK with that.
And I'm also pleased that the portrait is
bleached, because that suits my mood.

Fig. 2. Example commentary by The Painting Fool, from the *You Can't Know my Mind* exhibition, Paris, June 2013.

Managing the expectations and perceptions of the observers was a key aspect of this project. To this end, we hung posters describing the behaviour of the software as exhibiting aspects of intentionality, imagination, skill, appreciation, reflection and learning (six of the essential behaviours described above). Moreover, the software's actions and output were tailored to support the perception of these behaviours and an impression of creativity in the software by observers present in the exhibition, especially those sitting for a portrait.

Portraits were painted with people sitting in front of a laptop. It was immediately made clear that (i) the software was modelling a 'mood' to direct its painting, and (ii) the sitter was very much a tool for the software, not the other way around. This was achieved by opening remarks from the software such as: "Thank you for being my model. I'm in a negative mood right now, so I would like you to express a sad emotion." This was followed by The Painting Fool explicitly directing the sitter, while video recording them. A still image was then extracted where the sitter was expressing an emotion. Machine vision techniques were applied to remove the background, into which was substituted one of 1,000 abstract art images, to which one of 1,000 image filters was applied. The filter was chosen to increase the chances that the resulting image might reflect a changing simulated mood gained through reading newspaper articles, as described in [10]. The same filter was applied to the face of the sitter placed in the foreground, producing in a few seconds an image conception, or sketch for the portrait, such as the first image of figure 2.

Following this, a canvas appeared on screen, and a hand holding either a pencil, paint brush or pastel stick made virtual marks on the canvas leading to a non-photorealistic rendering of the background and foreground of the portrait, taking between 2 and 10 minutes, depending on the style. An example portrait is given at the bottom of figure 2, which was printed and given to the sitter, along with the commentary (the whole of figure 2). The most important aspect of the commentary is the expression of intention, by first showing a conception of the type of portrait the software aimed to produced, then showing what it produced and finally analysing and criticising – using machine vision techniques described in [44] – its results with respect to its aims.

The purpose of the exhibition was cultural, not scientific, and no experimentation was undertaken. From our experience, however, we contend that the behaviours exhibited by the software and explained in poster form enabled people to be surprised by the resulting portrait (and many of the 100 or so sitters in the exhibition were very surprised), while still projecting creativity onto the software. This upheld the aim of the *You Can't Know my Mind* exhibition: as it used some intelligence, and could explain its actions, it was somewhat appropriate to employ the word 'mind' with reference to The Painting Fool. However, as the process was unpredictable due to the dynamic nature of the software's changing mood, it was impossible to know this mind, and people realised that some software is written not to be a tool, but to be a creative individual. In fact, when in the most negative of moods, The Painting Fool refused to paint a portrait and sent the (often shocked) sitter away, citing a particularly depressing keyphrase in a particularly distressing newspaper article that it had recently read.

In these cases, The Painting Fool pointed out explicitly: "No random numbers were used in coming to this decision". This is because we feel that *accountable unpredictability* is important for creative systems. That is, we have found that when people realise that a certain important event has happened or an important artefact has been produced because of a random act, any dialogue (perceived or real) comes to an abrupt halt, and detracts from the creative experience. In contrast, unpredictability through accountable actions such as reading newspaper articles can add a great deal to a creative experience, at the very least by providing additional talking points.

Hypothesis 5 *Accountable unpredictability enhances the experience people have when told about software creating an artefact, whereas random number based unpredictability detracts from the experience.*

5 Formally Capturing Progress in Creative Systems

Naturally, another major set of stakeholders in the notion of software being creative are the Computational Creativity researchers who aim to write such systems, and use them to study creativity in people and machines. As they are familiar with the issues of simplistic arguments for and against creativity in software, these stakeholders require more formalism in any argumentation put forward to support the hypothesis of increased creativity in software.

We have focused on formalising the general notion of *progress* in Computational Creativity research. To do this, we first introduced the FACE and IDEA descriptive models in [19] and [45]. The FACE model categorises generative acts by software into those at (g)round level, during which base objects are produced, and (p)rocess level, during which methods for generating base objects are produced. These levels are sub-divided by the types of objects/processes they produce: F_g denotes a generative act producing some framing information, A_g denotes an act producing an aesthetic measure, C_g denotes an act producing a concept and E_g denotes an act producing an example of a concept. Generative acts producing new processes are defined accordingly as F_p, A_p, C_p and E_p. Tuples of generative acts are collated as *creative acts*, and various calculations and recommendations are suggested in the model with which to compare creative systems. We developed the IDEA model so that creative acts and any impact they might have could be properly separated. We defined various stages of software development and used an ideal audience notion, where people are able to quantify changes in well-being and the cognitive work required to appreciate a creative act and the resulting artefact and/or process.

The majority of researchers develop software using only themselves as an evaluator, because observer-based models are too time-consuming to use on a day-to-day basis. These informal in-house evaluation techniques generally do not capture the global aims of the research project, or of the field (e.g., producing culturally important artefacts and/or convincing people that software is acting in a creative fashion). This can lead to situations where systems are presented as feats of engineering, with little or no evaluation at all [34]. In [20], we argue that assessing *progress* is inherently a process-based problem, and hence we concentrate our formalism on processes, tempered with aspects of artefact evaluation. In the subsections below, we present this formalism with worked examples, followed by a case study describing the development of an evolutionary art system.

5.1 Formal Assessment of Progress

We combine the most useful aspects of the IDEA and FACE models, the list of essential behaviours described in section 4.1, and certain aspects of assessing artefact value in a diagrammatic formalism for evaluating progress in the building of creative systems. We focus on the creative acts that software performs, the artefacts it produces and the way in which audiences perceive it and consume its output. We simplify by assuming a development model where a single person or team develops the software, with various major points where the program is sufficiently different for comparisons with previous versions. We aim for the formalism to be used on a daily basis without audience evaluations, to determine short term progress, but for it also to enable fuller audience-level evaluations at the major development points. We also aim for the formalism to help determine progress in projects where there are both *weak* and *strong* objectives, focused, respectively, on the production of increasingly higher valued artefacts, and on increasing the perception of creativity people have of the system. We found that the original FACE model didn't enable us to properly express the process of building and executing generative software. Hence another consideration for our formalism is that it can capture various timelines both in the development and the running of software in such a way that it is fairly obvious where the programmer contributed creatively and where the software did likewise.

5.2 Diagrammatic Capture of Timelines

Taking a realistic but abstracted view of generative software development and deployment, we identify four types of timeline. Firstly, generative programs are developed in **system epochs**, with new versions being regularly signed off. Secondly, each process a program undertakes will have been implemented during a **development period** where creative acts by programmer and program have interplayed. Thirdly, at run-time, data will be passed from process to process in a series of creative and administrative **subprocesses** performed by software and programmer. Finally, each subprocess will comprise a sequence of generative or administrative **acts**.

We capture these timelines diagrammatically: the four different kinds of transitions are highlighted with coloured arrows in figure 3(a). The blue arrow from box α to β represents a change in epoch at system level. The red arrows overlapping a *process stack* represent causal development periods. The green arrows represent data being passed from one subprocess to another at run-time. The brown arrows represent a series of generative/administrative acts which occur within a subprocess. Inside each subprocess box is either a $<$ creative act $>$ from the FACE model (i.e., a sequence of generative acts), or an [administrative act] which doesn't introduce any new concept, example, aesthetic or framing information/method. Administrative acts were not originally described in the FACE model, but we needed them to describe certain progressions during software development. For our purposes here, we use only T to describe a translation administrative act often involving programming, and S to describe when an aesthetic measure is used to select the best from a set of artefacts. We employ the FACE model usage of lower-case letters to denote the output from the corresponding upper-case generative acts. Furthermore, we extend the FACE notion of (g)round and (p)rocess level generative acts with (m)eta level acts during which process generation methods are invented. As in the original description of the FACE model, we use a bar notation to indicate that a particular act was undertaken by the programmer. We use a superscripted asterisk (*) to point out repetition.

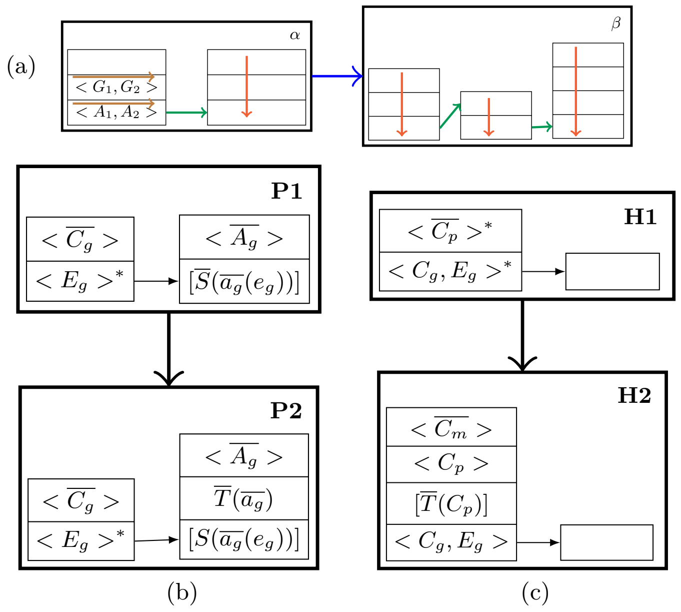


Fig. 3. (a) Key showing four types of timelines (b) progression of a poetry system (c) progression of the HR system.

As a simple example diagram, figure 3(b) shows the progression from poetry generator version P1 to P2. In the first version, there are two process stacks, hence the system works in two stages. In the first, the software produces some example poems, and in the second the user chooses one of the poems (to print out, say). The first stack represents two timesteps in development, namely that (a) the programmer had a creative act $< \overline{C_g} >$ whereby he/she came up with a concept in the form of some code to generate poems, and (b) the software was run to produce poems in creative acts of the form $< E_g >^*$. The second stack represents the user coming up with an idea for an aesthetic, e.g., preferring lots of rhyming, in creative act $< \overline{A_g} >$, and then applying that aesthetic $\overline{a_g}$ him/herself to the examples produced by the software, in the *selection* administrative act $[\overline{S}(\overline{a_g}(e_g))]$, which maps the aesthetic $\overline{a_g} : \{e_g\} \rightarrow [0, 1]$ over the generated examples, and picks the best one. In the $P2$ version of the software, the programmer undertakes the *translation* act $[\overline{T}(\overline{a_g})]$, writing code that allows the program to apply the rhyming aesthetic itself, which it does at the bottom of the second stack in box $P2$.

Figure 3(c) shows a progression in the HR automated theory formation system [12] which took the software to a meta-level, as described in [11]. HR operates by applying production rules which invent concepts that categorise and describe input data. Each production rule was invented by the programmer during creative acts of the type $< \overline{C_p} >$, then at run-time, HR uses the production rules to invent concepts and examples of them in $< C_g, E_g >^*$ acts. In the meta-HR version, during the $< \overline{C_m} >$ creative act, the programmer had the idea of getting HR to form theories about theories, and in doing so, generate concept-invention processes (production rules) in acts of the form $< C_p >$. The programmer took meta-HR's output and translated it via $[\overline{T}(C_p)]$ into an implemented production rule that HR could use, which it does at the bottom of the stack in box H2.

5.3 Comparing Diagrams and Output

Examining the transition from one epoch-level diagram to another should provide some shortcuts to estimate audience reactions, especially when these are linked to strong objectives. As with the original FACE model, the diagrams make it obvious where creative or administrative responsibility has been handed over to software, namely where an act which used to be barred has become unbarred, i.e., the same type of generative act still occurs, but it is now performed by software rather than programmer. For instance, this happened when the $\overline{S}$ became an S in figure 3(b) and when the $\overline{C_p}$ became a C_p in figure 3(c). At the very least in these cases, an unbiased observer would be expected to project more autonomy onto the software, and so progress in the strong sense has likely happened. In addition, the diagrams make it obvious when software is doing more processing in the sense of having more stacks, bigger stacks or larger tuples of acts in the stack entries. Moreover, the diagrams make it clear when more varied or higher-level creative acts are being performed by the software. All of these features have the potential to convince audience members that software is being more sophisticated, and can be taken as a preliminary indicator of progress.

When dealing with actual external evaluation, where people don't know what the software does, we suggest that the diagrams above (or verbalisations/simplifications of them) can be used to describe the software to audience members, to explain what the software does, and what the programmer has done in the project. Audience members can then be asked whether they would project any of the essential behaviours from section 4.1 onto any of the creative acts undertaken by the programmer or by the system. Thus, one method for estimating progress from version $v1$ of a creative system to version $v2$ that takes into account features of both processing features and artefact quality would be:

- show audience members the diagrams for $v1$ and $v2$ as above, and explain the acts undertaken by the software, then
- show audience members the output from $v1$ and $v2$, and,
- ask each person to compare the pair of product and process for $v1$ with that of $v2$.

A statistical analysis could then be used to see whether the audience as a whole evaluates the output as being better, worse or the same, and whether they think that the processing is better, worse or the same in terms of the software seeming less uncreative.

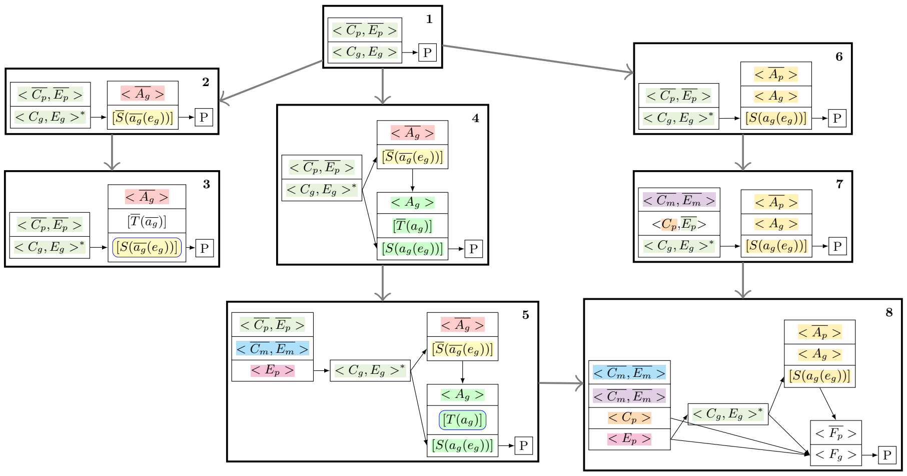


Fig. 4. The progression of an evolutionary art program through eight system epochs, taken from [20].

5.4 A Case study in Evolutionary Art

Evolutionary art – where software is evolved which can generate abstract art – has been much studied within Computational Creativity circles [53]. Based on actual projects which we reference, we hypothesise here the various timelines of progress that could lead from a system with barely any autonomy to one with nearly full autonomy. Figure 4 uses our diagrammatic approach to capture three major lines of development, with the final (hypothetical) system in box 8 representing finality, in the strong sense that the software can do very little more creatively in generating abstract art. Since features from earlier system epochs are often present in later ones, we have colour-coded individual creative acts as they are introduced, so the reader can follow their usage through the systems. If an element repeats with a slight variation (such as the removal of a bar), this is highlighted. Table 1 is a key to the figure, which describes the most important creative and administrative acts in the systems. Elements in the key are indexed with a dot notation: *system.process-stack.subprocess* (by number, from left to right, and top to bottom, respectively). System diagrams have repetitive elements, so that the timelines leading to its construction and what it does at run-time can be read in a stand-alone fashion.

Following the first line of development, system 1 of figure 4 represents an entry point for many evolutionary art systems: the programmer invents ($\overline{C_p}$) (or borrows) the concept formation process of crossing over sets of mathematical functions to produce offspring sets. He/she also has an idea ($\overline{E_p}$) for a *wrapper* routine which can use such a set of functions to produce images. He/she then uses the program to generate (C_g) a set of functions and employ the wrapper to produce (E_g) an image which is sent to the (P)rinter. The crossover and subsequent image generation is repeated multiple times in system 2, and then

ID	**Event**	**Explanation**
1.1.1	$\overline{C}_p$	The programmer invents the idea of crossing over two sets of mathematical functions to produce a new set of mathematical functions.
1.1.1	$\overline{E}_p$	The programmer implements a wrapper method that takes a set of mathematical functions and applies them to each (x, y) co-ordinate in an image to produce an RGB colour.
1.1.2	C_g	The software generates a new set of functions by crossing over two pairs of functions.
1.1.2	E_g	The software applies these functions to the (x, y) co-ordinates of an image, to produce a piece of abstract art.
2.2.1	$\overline{A}_g$	The programmer had in mind a particular aesthetic (symmetry) for the images.
2.2.2	$\overline{S}(\overline{a_g}(e_g))$	The programmer uses his/her aesthetic to select an image for printing.
3.2.2	$\overline{T}(\overline{a_g})$	The programmer took their aesthetic and turned it into code that can calculate a value for images.
3.2.3	$\boxed{S(\overline{a_g}(e_g))}$	The software applies the aesthetic to select one of a set of images produced by the software.
4.3.1	A_g	The software uses machine learning techniques to approximate the programmer's aesthetic.
4.3.2	$\overline{T}(a_g)$	The programmer hand-translates the learned aesthetic into code.
4.3.3	$S(a_g(e_g))$	The software applies the new aesthetic to choosing the best image from those produced.
5.1.2	$\overline{C}_m$	The programmer has the idea of getting the software to search through a space of wrapper routines.
5.1.2	$\overline{E}_m$	The programmer implements this idea.
5.1.3	E_p	The software invents a new wrapper.
5.4.2	$\boxed{T(a_g)}$	The software translates the machine-learned aesthetic itself into code.
6.2.1	$\overline{A}_p$	The programmer has the idea of getting the software to invent a mathematical fitness function.
6.2.2	A_g	The software invents a novel aesthetic function.
6.2.3	$S(a_g(e_g))$	The software selects the best image according this aesthetic.
7.1.1	$\overline{C}_m$	The programmer has the idea of getting the software to invent and utilise new function combination techniques, generalising crossover.
7.1.1	$\overline{E}_m$	The programmer implements this idea so that the software can invent new combination techniques.
7.1.2	C_p	The software invents a novel combination technique.
8.4.1	$\overline{F}_p$	The programmer has the idea of getting the software to produce a commentary on its processes and the images it produces.
8.4.2	F_g	The software produces a commentary about its process and product.

Table 1. Key to figure 4.

the programmer – who has invented ($\overline{A_g}$) their own aesthetic – chooses a single image to print. In system 3, as in the poetry example above, the programmer translates their aesthetic into code so the program can select images. This is a development similar to that for the NEvAr system [41].

Following the second line of development, in system 4, the programmer selects multiple images using his/her own aesthetic preferences, and these become the positives for a machine learning exercise as in [39]. This enables the automatic invention (A_g) of an aesthetic function, which the programmer translates by hand $\overline{T}(a_g)$ from the machine learning system into the software, as in [15], so the program can employ the aesthetic without user intervention. In system 5, more automation is added, with the programmer implementing their idea ($\overline{C_m}$) of getting the software to search for wrappers, then implementing this ($\overline{E_m}$), so that the software can invent (E_p) new example generation processes for the system.

Following the final line of development, in system 6, we return to aesthetic generation. Here the programmer has the idea ($\overline{A_p}$) of getting software to mathematically invent fitness functions, as we did in [13] for scene generation, using the HR system [12] together with The Painting Fool [16]. In system 7, the programmer realises ($\overline{C_m}$) that crossover is just one way to combine sets of functions, and gives ($\overline{E_m}$) the software the ability to search a space of combination methods (C_p). The software does this, and uses the existing wrapper to turn the functions into images. System 8 is the end of the line for the development of the software, as it brings together all the innovations of previous systems. The software invents aesthetic functions, innovates with new concept formation methods that combine mathematical functions, and generates new wrappers which turn the functions into images. Finally, the programmer has the idea ($\overline{F_p}$) of getting the software to write commentaries, as in [18], about its processing and its results, which it does in generative act F_g.

Tracking how the system diagrams change can be used to estimate how audiences might evaluate the change in processing of the software, in terms of the extended creativity tripod described above. Intuitively, each system represents progress from the one preceding it, justified as follows:

1 $\rightarrow$ **2**: $< C_g, E_g > \rightarrow < C_g, E_g >^*$

Simple repetition means that the software has more *skill*, and the introduction of independent user selection shouldn't change perceptions about *autonomy*.

2 $\rightarrow$ **3**: $\overline{S} \rightarrow S$

By reducing user intervention in choosing images, the software should appear to have more *skill* and *autonomy*.

1 $\rightarrow$ **4**: Introduction of A_g and $S(a_g(e_g))$ acts

Machine learning enables the generation of novel aesthetics (albeit derived from human choices), which should increase perception of *innovation*, *appreciation* and *learning*, involving more varied creative acts.

4 → 5: Introduction of an E_p act, $\overline{T} \rightarrow T$

Wrapper generation increases the variety of creative acts, and may increase perception of *skill* and *imagination.*

1 → 6: Introduction of A_g and $S(a_g(e_g))$ acts

The software has more variety of creative acts, and the invention and deployment of its own aesthetic – this time, without any programmer intervention – should increase perception of *intentionality* in the software.

6 → 7: Introduction of a C_p act

Changes in the evolutionary processes should increase perceptions of *innovation* and *autonomy.*

5, 7 → 8: Introduction of an F_g act

Framing its work should increase perceptions of *accountability* and *reflection.*

With all strands brought together, the programmer does nothing at run-time and can contribute little more at design time. The software exhibits behaviours onto which we can meaningfully project words like skill, appreciation, innovation, intentionality, reflection, accountability and learning, which should raise impressions of autonomy, and make it difficult to project uncreativity onto the software.

Hypothesis 6 *The diagrammatic formalism given above – or some extension of it – is sufficient to capture the creative acts performed in building and running any kind of generative software. Moreover, when this is used alongside audience evaluation of the artefacts produced, a formal assessment of progress in creative software development can be achieved.*

6 Software as part of a Creative Community

For each domain in which creative software operates, there is a community of people who have a stake in the notion of whether software working in that domain is perceived as creative. As described in this section, we have recently started to embed our software in such a community, for various reasons, including the study of how people react to it and to the work that it produces. These experiments will form part of a larger study of how people accept (or not) creative technologies that undertake activities which used to be the purview of people only.

6.1 Accountable Subjectivity

Applying aesthetic judgements and expressing preferences are important kinds of activity that contribute to the perception of a person or piece of software as being creative. Aesthetics and preferences allow a creative entity, be it a person or software, to express founded judgement (even if we regard the judgement as worthless, or subjectively 'wrong') on creative artefacts, both those created by the entity itself and those created by others. It can also serve as a driving force behind future creation, allowing someone to work towards goals that they have set themselves and strengthening claims of intentionality.

Despite this, little work has been done to build systems which can generate aesthetic preferences of their own and apply them intelligently. One reason for this may be the uncomfortable clash between the subjective and the objective that so often affects research in Computational Creativity. The notion of 'optimality' in many creative domains, particularly those associated with the arts, is a contentious one and leads to much criticism of systems which attempt to quantify the quality of an artefact. The idea of having a system quantify the quality of an *opinion* on creative artefacts is equally controversial, if not more so. Similarly, in the past, the question of how to quantify the degree to which a system is creative was also a subjective and controversial task. In this case, researchers such as Ritchie found it useful to use metrics which dealt with abstract notions of creativity without directly laying out objective measures of quality for any particular artefact or medium. Ritchie's criteria are described in [52], and have been used in many evaluations of creative systems in a variety of different fields and media.

We propose here a similar set of criteria which apply to aesthetics or preferences rather than creative systems. By using abstract metrics, we can avoid talking about aesthetic measures in objective ways, while retaining a meaningful vocabulary with which to describe different kinds of aesthetics. These metrics can be used to evaluate *aesthetic comparator functions*, namely binary functions which take two examples of a type of object, and then return -1, 0 or 1 depending on whether the first object is preferred less, the same as, or more than the second object. Assuming we have an aesthetic function f, and a set of objects the function expresses a preference over, O, we define the following criteria which can be used to differentiate aesthetic functions from one another. Note that these metrics do not necessarily represent a linear gradient of quality – different types of aesthetic function may be desirable in different scenarios.

The first metric is *specificity*. Specificity captures the degree to which the aesthetic represents a *total order* over the set of objects O. If an aesthetic can offer a definite preference (that is, a nonzero result) for many of the objects, it will have a high specificity, and vice versa. High-specificity aesthetics might suggest the aesthetic is experienced or well-developed in some way, if it is able to make clear distinctions between many different artefacts.

The second metric is *transitive consistency*. This captures how self contradictory the aesthetic function is. Suppose we have three artefacts: A, B and C, and our function f. We can write $A < B$ to indicate that B is preferred to A. We might expect that if $A < B$ and $B < C$ then $A < C$. Transitive consistency measures what proportion of O this holds for. In some scenarios, we might want a high transitive consistency, as this indicates a lack of contradictions in the preferences being expressed. However, in some scenarios, preferences can be complex and multi-objective, and it might be the case that transitivity does not hold for highly subjective opinions about artefacts produced by creative acts.

The third metric is *agreement*. Instead of being expressed in terms of a single aesthetic function, agreement is expressed about two different aesthetics, which we can call f and g. Agreement measures the proportion of the object set O that f and g agree on. This can be *strict*, in which case f and g must return exactly the same value for two objects to be said to agree. Alternatively, agreement can be *non-strict*, in which case f and g can either return the same value, or one of the functions can return zero (no preference) to be said to agree. Informally, agreement lets us assess how closely two aesthetic functions are aligned with each other. Of course, they may be in close agreement for very different reasons – this metric simply establishes similarity in the result of the subjective judgements.

Hypothesis 7 *The perception of creativity in software which produces artefacts within a creative community will be increased if the software can exhibit subjective judgements about its own work and that of others, and defend those judgements in an accountable way. This can be seen as part of a bigger picture of software exhibiting a personality, in order to be accepted into a creative community.*

6.2 A Case Study in Automatically Designed Videogames

A *game jam* is a contest where entrants attempt to make a videogame from scratch in a short period of time, normally with the added restriction of a theme which developers must incorporate into their game somehow. Ludum Dare is one of the largest regularly occurring game jams in the game development community, taking place three times a year and garnering over 2,000 entries in December 2013, where developers were given the theme 'You Only Get One'. The ANGELINA system is an automated videogame designer developed to investigate issues surrounding Computational Creativity in a ludic and interactive context [22]. Many different versions of ANGELINA have been developed, working with various different kinds of game, technologies and user guidance. The most recent iteration, ANGELINA-5, was designed to enter game jams, by allowing it to be given just a theme in plain text as a starting point. This theme is then interpreted by ANGELINA-5 and used to influence the design of the game.

ANGELINA-5 entered Ludum Dare for the first time in December 2013, the 28th edition of the event. One of the objectives was to investigate the reactions of various groups of people to a piece of creative software entering such a contest. To gain more insight into these groups, we entered two games designed by ANGELINA-5 to Ludum Dare 28. In the first submission, *To That Sect*[3], we included a commentary generated by ANGELINA-5 to illustrate the actions of the system, as well as multiple paragraphs describing the research behind ANGELINA-5 and identifying the game as the creation of a piece of software. In the second submission[4], we anonymised ANGELINA-5's commentary to remove references to it being software-based, edited it for grammar, and added

[3] *To That Sect* game: `www.tinyurl.com/tothatsect`

[4] *Stretch Bouquet Point* game: `www.tinyurl.com/stretchpoint`

	To That Sect	Stretch Bouquet Point	Jet Force Gemini
Overall	36	29	23
Fun	34	30	26
Audio	73	43	74
Graphics	43	33	36
Mood	77	39	80
Innovation	64	33	59
Theme	32	30	26
Humour	48	59	51

Table 1. Percentile rankings for ANGELINA-5's two games entered into Ludum Dare 28, and its single entry to Ludum Dare 29 *(Jet Force Gemini)*. Note that higher percentile rankings indicate higher achievements. There were 780 submissions in the LD28 track, and 1004 entries in the LD29 track.

no supplementary explanation about the software, the origin of the game, or anything to connect the game with a digital author. The ratings process for Ludum Dare takes place in the 22 days following the contest, and is conducted as a peer review system, where each entrant is asked to rate and review games by other entrants. Ratings are given as marks out of five for eight categories: Audio, Graphics, Mood, Theme, Humour, Fun, Innovation and Overall.

The results for the two entries by ANGELINA-5 can be seen in table 1. While we were unable to get specific vote data, we do know that 70 people rated *To That Sect*, the non-anonymised submission, while 26 people rated *Stretch Bouquet Point*[5]. While it is impossible to calculate confidence intervals for these ratings without the vote data, we can see that they differ by hundreds of positions for some categories such as Mood and Audio. We can also see a noticeable difference in the comments left by some of the reviewers underneath both submissions, in terms of their tone and attitude when dealing with each game. Many commentators indirectly criticise the anonymised game, with comments such as "You made me feel something there. Don't make me put it into words though". Other commentators made more obvious statements of criticism or praise, such as "This was a rather annoying experience" or "This game feels dreamy. The audio is intense." Only one comment included both praise and criticism. We attribute the indirect or sarcastic comments to an unwillingness to potentially criticise a person for performing poorly, even though other reviewers were less tactful. Ludum Dare is often used as a learning experience for amateur developers, and many children enter using simple game creation tools. We believe many reviewers felt uncomfortable with direct criticism for this reason.

By contrast, comments on *To That Sect* were more balanced in nature, often offering both praise and criticism in equal amounts, e.g., "Angelina seems really good at creating an atmosphere with both sound and visuals. But the game

[5] This is due in part to ANGELINA-5's small following on the internet, which promoted the non-anonymised submission more than normal.

part of it seems a bit lacking still". In the description of the game, we asked people to rate it as they would any other Ludum Dare entry, hoping to dissuade people from reviewing the *concept* of ANGELINA-5 rather than the game itself. Nevertheless, many reviewers suggest that their scores were influenced by their appraisal of ANGELINA-5 as a novel system, rather than what it was capable of creating, e.g., "creating a program to create your game ... [is] certainly not something you see every day. On that front alone, this gets a lot of points for innovation". These results suggest that reviewers were unable to separate the creator from the artefact, and were incapable of reviewing the game as if created by a person. For instance, *To That Sect* rated 282nd of 780 for Innovation. These ratings are subjective, and it is hard for us to objectively assess them. However, we do not believe there is anything particularly innovative about *To That Sect*. As such, we must attribute this high ranking to reviewers assessing the game as a product of ANGELINA-5. It seems that reviewers projected (human) innovation in the ANGELINA project onto the game it produced.

We can compare the results of ANGELINA-5's debut in Ludum Dare with the results garnered from a second entry to the game jam in April 2014, Ludum Dare 29. This time ANGELINA-5 was only entered into the game jam once, with the game *Jet Force Gemini*, created in response to the theme *Beneath The Surface*. As with the non-anonymised entry in Ludum Dare 28, *Jet Force Gemini* was entered with a commentary describing some of the decisions contributing to the design process. Table 1's rightmost column shows the results for *Jet Force Gemini* in contrast to the entries in Ludum Dare 28. The number of entries in Ludum Dare 29 was nearly 30% larger than Ludum Dare 28; ANGELINA's percentile scores drop for four of seven specialised categories, and fall dramatically in the *Overall* rating.

We believe this is evidence of the relationship between the observers and ANGELINA shifting over time. While some of the comments underneath *Jet Force Gemini* indicate that the reviewer is encountering ANGELINA-5 for the first time (which is unsurprising, since the number of reviewers account for less than 1% of total Ludum Dare entrants) others explicitly note that they are reviewing ANGELINA-5's games for a second time. One states that 'I'm sorry to say that I can't really see improvements from last time', indicating that there is either an expectation of growth on the part of the software, or an expectation that the software's author will grow the software over time. Despite many of the other comments being generally positive, the drop in ratings suggests that people perhaps feel less compelled to rate ANGELINA-5 highly for novelty value alone. Given that Ludum Dare is a community built on the idea of improving creative skills through regular practice, it is interesting to note the expectation of growth shown by some reviewers. We hypothesise that this may be a factor which is particularly important for creative individuals in assessing creativity, as opposed to other types of observer.

We can also examine reactions to particular elements of ANGELINA-5's work and compare it to critiques of similar games. One comment on *To That Sect* states "If it [had] added shooting at the statues that you must avoid and a goal how much ships you have to collect, it would have been better. It felt like playing [an] ‘art-message’ type of game”. *LITH*[6] is a game entered into the competition by a human designer, where the player navigates a maze and collects bags of gold coins, while avoiding patrolling robots. They can escape to an exit at any stage, with their score being the amount of gold collected. While not an exact duplicate, the rules of *LITH* are very similar to those of *To That Sect*, i.e., search for as many objects of a certain type as possible, while avoiding another object, then exit. *LITH* was entered in the same track as ANGELINA-5’s games, and ranked 95th Overall, 125th for Fun, and 274th for Theme. None of the comments on *LITH* reference the game’s rulesets in a critical way. Notably, *LITH* ranks 259 places above *To That Sect* for Theme. This is significant, as the *LITH* designer justifies its theme in a fairly thin way, by saying simply that the player only has one opportunity to save their score (which they do by ending the game, as in *To That Sect*). The games are by no means identical: *LITH*’s level is more closed in to accentuate a feeling of claustrophobia, but the similarities are many. This analysis suggests a fundamental difference in how people evaluate a game when they have knowledge and when they have no knowledge of its designer and design process.

Hypothesis 8 *There can be both positive and negative biases at work when people consume artefacts in the knowledge that computers created them. By managing both cases in a creative community context, we can increase perception of software as being creative and enjoyment of the artefacts produced. This increase will be further fuelled if the software shows clear growth in sophistication in the field, and expresses this through its processes and products.*

7 Conclusions and Future Work

Simply stated, one of the main aims of research into Computational Creativity is to one day see creative software properly embedded into society. To achieve this aim, larger sectors of society need to join the effort, including creative communities within the arts and sciences, the creative industries, technology firms, and the next generation of Artificial Intelligence researchers. Hence, we need to convince certain sets of stakeholders that creative software is no fantasy, but a potential reality that will bring benefits to society. As described above, we have studied three sets of stakeholders, namely the general public, fellow Computational Creativity researchers, and a specific community of creative people, namely videogame designers. These studies have enabled us to make concrete hypotheses related to how stakeholder communities perceive creativity in software, and how best to manage that perception in the future. Based on our immersion in the stakeholder communities mentioned, we have argued above in favour of the truth of the hypotheses, with extended discourse and argumentation given in [17] and [20] amongst other papers. We believe it is now time to turn the

[6] *LITH* game: `www.tinyurl.com/lith-ludum`

hypotheses into experiments designed to see whether the ways in which sets of stakeholders perceive and react to creative software fit our beliefs.

Our first hypothesis is pitched somewhat at a meta-level, in that it proposes that different stakeholder groups see creative systems differently and their perception of software behaviour could and should be managed in a bespoke manner. We can therefore imagine an experiment where we present the processes and products from creative software to different stakeholder groups and assess their reaction to see if there is indeed a difference in how different groups react, learning from analyses of the results. Hypothesis 2 encompasses much of our philosophical position on the notion of creativity being essentially contested and secondary in nature. One can imagine restricting participants in an experiment to fairly constrained groups, and testing whether there is general (healthy) disagreement about the nature of creativity in people and software or not, and further testing whether there is more consensus about software being uncreative. To properly test hypothesis 2, we would need to ask participants about the essential behaviours – such as intentionality, learning and reflection – they perceive to be taking place in software and see how it affects their perception of uncreativity in the system.

Our third hypothesis makes a bold statement: that blind comparison tests damage the long-term goal of embedding creative software in society, by emphasising the evident humanity gap. If this effect is true, it would be borne out by a Turing-style test where, when people are told that it was software that produced an artefact that they particularly liked, they were also asked about whether their perception of the creative act and/or the artefact had changed in light of the new knowledge. More pointed questions about the nature of any change in perception could lead to insights about how to manage the humanity gap in future projects. This would lead into an experiment to address hypothesis 4, where computer generated artefacts were presented as re-imagined pieces with specific management of the relative lack of humanity in the generation of the artefacts. The re-imagining would specifically include commentaries and other framing information produced by the creative system. If hypothesis 4 is correct, people would appreciate the re-imagined versions of artefacts more than those presented merely as computer-generated versions from the human oeuvre.

By proposing that random number generation detracts from an experience of a creative act, whereas more accountable unpredictability can benefit the experience, hypothesis 5 is more specific than those preceding it. We can imagine an experiment where one set of participants are told that a particularly impressive creative act (in terms of the processing performed and/or the resulting artefacts) was because of a random event, and another set are given interesting framing information about what led – in a non-random way – to the same unpredictably good creative act. If the latter group appreciated the creative act and its results more than the former group, the truth of the hypothesis would be upheld.

We have already started work on testing hypothesis 6, i.e., that the formalism presented in [20], can capture notions of progress when building creative systems. That is, we have used the formalism to capture abstracted timelines leading to the building of certain creative systems, and timelines where that software operates and produces artefacts of value. However, to convince the Computational Creativity researcher stakeholders of the value of the formalism, we need to work with them to capture the essence of their approaches to implementing and operating creative software. Moreover, our audience evaluation model is far from complete. We plan to employ the criteria specified in [52], for more fine-grained evaluations of the quality, novelty and typicality of artefacts. We will also import audience reflection evaluation schemes from the IDEA descriptive model, e.g., change in well-being, cognitive effort and emotional responses such as surprise and amusement.

The final two hypotheses we present above relate to communities of creative people into which creative software is implanted. To address hypothesis 7, we will need to implement software behaviours which can meaningfully be described as subjective, and we plan to do so with the ANGELINA videogame generation system, and others such as The Painting Fool automated artist. With such systems, we can experiment to see whether members of the creative community are more impressed by subjective software or not. Such an experiment could be simultaneously used to address the final hypothesis, with knowledge of the computational origins of artefacts systematically withheld in order to see whether positive or negative biases hold in different creative communities. Similarly, experiments where participants are told about the intellectual growth of a system could be carried out, to see if this influences their impression of the software. An analysis of the findings from such experiments could help pave the way for software to be full members of these kinds of communities.

Looking at the three stakeholder groups studied here, we see some emerging generalities. In particular, looking at behaviours where systems exhibit subjectivity and intentionality, it seems clear that in all three groups, *personality modelling* in software has the potential to increase the impression that people have of what software does and, in turn, what it produces. This is part of a new understanding of creative acts as being potentially interesting, even dramatic, episodes of activity which can amuse and engage people, rather than a means to the end of producing an artefact of value. This is in contrast with the traditional idea that the value of the output from software can increase people's appreciation of the creativity it exhibits. While the traditional view is often correct, it is not the only model of managing perceptions of creativity in software.

The hypotheses presented here are only a subset of those which should be proposed and addressed in the future of Computational Creativity research. Not addressing such issues would be a mistake, as stakeholder perception of creativity in software will in part dictate the number of researchers and businesses coming into the field. Done badly, handling of stakeholder perceptions could stall the forward progress achieved towards embedding creative software in society. As a recent controversial example, online retailer Amazon briefly sold T-shirts with slogans such as *"Keep Calm and Rape a Lot"* [42]. The T-shirt company responsible posted an apology on its website, and insisted that the offending articles were "automatically generated using a scripted computer process running against hundreds of thousands of dictionary words". This may be the first example of computer generated artefacts causing such offence and a company – while taking responsibility – blaming generative software for poor quality artefacts, while tacitly acknowledging that the software had taken on unsupervised creative responsibilities in their workplace.

Situations where software is employed independently for creative purposes in commerce and elsewhere are likely to become more commonplace in the future. As a more positive example, IBM researchers have recently undertaken research to explore the commercial potential of Computational Creativity [31], with particular emphasis on culinary creativity [48,55]. Creative software will make great inventions and make terrible mistakes in the future, and this will lead to a re-evaluation of humanity as being the centre of the creativity universe. Managing stakeholder perceptions of creativity in software will be paramount in making this transition as smooth and as fruitful for society as possible.

Acknowledgements

Some of the work presented here was originally explored in [17] and [20], and we are very grateful to the organisers of the AISB 2014 symposium on Computing and Philosophy, and the organisers of the 2014 International Conference on Computational Creativity. We wish to thank the many researchers with whom we have discussed the views presented in this chapter, especially our colleagues in the Computational Creativity group at Goldsmiths College. This research has been funded by EPSRC grants EP/L00206X and EP/J004049, and with the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant numbers: 611553 (COINVENT) and 611560 (WHIM).

References

1. National Science Foundation, CreativeIT, Program Solicitation 09-572 http://www.nsf.gov/pubs/2009/nsf09572/nsf09572.htm, 2009.
2. The Seventh Framework Programme (2007-2013) of the European Union: Information and Communication Technologies. http://cordis.europa.eu/fp7/ict/docs/ict-wp2013-10-7-2013-with-cover-issn.pdf, 2013.
3. M. A. Arbib and M. B. Hesse. *The Construction of Reality.* Cambridge University Press, 1986.
4. J. Langshaw Austin. *How to do things with words.* Oxford University Press, 1975. The William James Lectures delivered at Harvard University in 1955.
5. M. A. Boden. What is creativity? In M. A. Boden, editor, *Dimensions of Creativity*, pages 75 – 117. MIT Press, 1996.
6. M. A. Boden. *The Creative Mind: Myths and Mechanisms.* Weidenfield and Nicholson, 1990.
7. O. Bown. Empirically grounding the evaluation of creative systems: Incorporating interaction design. In *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.

8. A. Cardoso, T. Veale and G. A. Wiggins. Converging on the divergent: The history (and future) of the International Joint Workshops in Computational Creativity. *AI Magazine*, 30(3), 2009.
9. J. Charnley, A. Pease, and S. Colton. On the notion of framing in Computational Creativity. In *Proceedings of the Third International Conference on Computational Creativity*, 2012.
10. S. Colton and D. Ventura. You can't know my mind: A festival of Computational Creativity. In *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.
11. S. Colton. Experiments in meta-theory formation. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*, 2001.
12. S. Colton. Automated theory formation in pure mathematics. Springer. 2002.
13. S Colton. Automatic invention of fitness functions with application to scene generation. In *Proceedings of the EvoMusArt Workshop*, 2008.
14. S. Colton. Creativity versus the perception of creativity in computational systems. In *Proceedings of the AAAI Spring Symposium on Creative Intelligent Systems*, 2008.
15. S Colton. Evolving a library of artistic scene descriptors. In *Proceedings of the EvoMusArt Conference*, 2012.
16. S. Colton. The Painting Fool: Stories from building an automated painter. In J. McCormack and M. d'Inverno, editors, *Computers and creativity*, pages 3–38. Springer, 2012.
17. S. Colton, M. Cook, R. Hepworth, and A. Pease. On acid drops and teardrops: Observer issues in computational creativity. In *Proceedings of the 7th AISB Symposium on Computing and Philosophy*, 2014.
18. S. Colton, J. Goodwin, and T. Veale. Full FACE poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, 2012.
19. S. Colton, A. Pease, and J. Charnley. Computational Creativity Theory: The FACE and IDEA descriptive models. In *Proceedings of the Second International Conference on Computational Creativity*, 2011.
20. S. Colton, A. Pease, J. Corneli, M. Cook, and T. Llano. Assessing progress in building autonomously creative systems. In *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.
21. S. Colton and G. A. Wiggins. Computational Creativity: The final frontier? In *Proceedings of the European Conference on AI*, 2012.
22. M. Cook, S. Colton, and J. Gow. Automating game design in three dimensions. In *Proceedings of the AISB Symposium on AI and Games*, 2014.
23. D. Dennett. Three kinds of intentional psychology. In *Perspectives in the Philosophy of Language: A Concise Anthology*, pages 163–186. Broadview Press, 2000.
24. A. Eigenfeldt, A. Burnett, and P. Pasquier. Evaluating musical metacreation in a live performance context. In *Proceedings of the Third International Conference on Computational Creativity*, 2012.
25. A. Ellegard. *Darwin and the General Reader: The Reception of Darwin's Theory of Evolution in the British Periodical Press, 1859-1872.* University Of Chicago Press, 1990.
26. K. Evans. *The Stuckists: The First Modernist Art Group.* Victoria Press, 2000.
27. J. Fahnestock. *Rhetorical Figures in Science.* Oxford University Press, 1999.
28. M. Franzen, P. Weingart, and S. Rödder. Exploring the impact of science communication on scientific knowledge production: An introduction. In S. Rödder, M. Franzen, and P. Weingart, editors, *The Sciences' Media Connection – Public Communication and its Repercussions*, pages 3–16. Springer, 2012.

29. W. Gallie. Essentially Contested Concepts. *Proceedings of the Aristotelian Society*, 56:167–198, 1956.
30. J. N. Gray. On the contestability of social and political concepts. *Political Theory*, 5(3), 1977.
31. A. Jagmohan, Y. Li, N. Shao, A. Sheopuri, D. Wang, L. Varshney, and P. Huang. Exploring application domains for computational creativity. In *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.
32. C. Johnson. Is it time for computational creativity to grow up and be irresponsible? In *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.
33. J. Jones. Santa bought me a Playstation. But it's still not art. The Guardian. 7th January 2014.
34. A. K. Jordanous. *Evaluating Computational Creativity: A Standardised Procedure for Evaluating Creative Systems and its Application.* PhD thesis, Department of Informatics, University of Sussex, 2012.
35. G. Lakoff. Why it matters how we frame the environment. *Environmental Communication*, 4(1):70–81, 2010.
36. L. Lambourne. *The Aesthetic Movement.* Phaidon, 1996.
37. B. Latour. *Science in Action: How to Follow Scientists and Engineers Through Society*, Open University Press, 1987.
38. G. Lemaine, R. Macleod, M. Mulkay, and P. Weingar. Problems in the emergence of new disciplines. In G. Lemaine, R. Macleod, M. Mulkay, and P. Weingar, editors, *Perspectives on the Emergence of Scientific Disciplines*, pages 1–26. Maison des Sciences de l'Homme, 1976.
39. Y. Li, C. Hu, L. Minku, and H. Zuo. Learning aesthetic judgements in evolutionary art systems. *Genetic Programming and Evolvable Machines*, 14(3):315–337, 2013.
40. J. Locke. *An essay concerning human understanding.* Oxford University Press 1975.
41. P. Machado and A. Cardoso. All the truth about NEvAr. *Applied Intelligence*, 16(2):101–118, 2002.
42. T. McVeigh. Amazon acts to halt sales of 'Keep Calm and Rape' T-shirts. The Guardian. 2nd March, 2013.
43. D. Moffat and M. Kelly. An investigation into people's bias against computational creativity in music composition. In *Proceedings of the Third Joint Workshop on Computational Creativity*, 2006.
44. D. Norton, D. Heath, and D. Ventura. Finding creativity in an artificial artist. *The Journal of Creative Behavior*, 47(2):106–124, 2013.
45. A. Pease and S. Colton. Computational Creativity Theory: Inspirations behind the FACE and the IDEA models. In *Proceedings of the Second International Conference on Computational Creativity*, 2011.
46. A. Pease and S. Colton. On impact and evaluation in Computational Creativity: A discussion of the Turing Test and an alternative proposal. In *Proceedings of the AISB symposium on Computing and Philosophy*, 2011.
47. A. Pease, S. Colton, R. Ramezani, J. Charnley and K. Reed. A discussion on serendipity in creative systems. In *Proceedings of the Fourth International Conference on Computational Creativity*, 2012.
48. F. Pinel, L. Varshney, and D. Bhattacharjya. A Culinary Computational Creativity System. In *This edition*, 2014.
49. J. A. Plucker and M. C. Makel. Assessment of creativity. *The Cambridge handbook of creativity*, pages 48–73, 2010.

50. J. R. Ravetz. Usable knowledge, usable ignorance: Incomplete science with policy implications. *Knowledge: Creation, Diffusion, Utilization*, 9(1):86–116, 1987.
51. J. R. Ravetz. *Scientific Knowledge and its Social Problems*. Transaction Publishers, 1996.
52. G. Ritchie. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17(1):67–99, 2007.
53. J. Romero and P. Machado. *The art of artificial evolution: A handbook on evolutionary art and music*. Springer, 2008.
54. J. Searle. A taxonomy of illocutionary acts. In Keith Günderson, editor, *Language, mind, and knowledge*, volume 7. University of Minnesota Press, 1975.
55. N. Shao, P. Murali, and A. Sheopuri. New developments in culinary Computational Creativity. In *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.
56. K. Smith. Mutually contested concepts and their standard general use. *Journal of Classical Sociology*, 2(3):329–343, 2002.
57. S. H. Stocking and L. W. Holstein. Manufacturing doubt: Journalists' roles and the construction of ignorance in a scientific controversy. *Public Understanding of Science*, 18:23–42, 2009.
58. K. Stuart. Video games and art: Why does the media get it so wrong? The Guardian. 8th January 2014.
59. A. M. Turing. Computing machinery and intelligence. *Mind*, vol. 59, pages 433–460, 1950.
60. W. K. Wimsatt. *The verbal icon: Studies in the meaning of poetry.* Number 123. University Press of Kentucky, 1954.