Values in Design: Theory and Practice[1]

DRAFT - April 24 2005 - DRAFT

Mary Flanagan, Hunter College

Daniel Howe, New York University

Helen Nissenbaum, New York University

Introduction

The idea that values may be embodied in technical systems and devices (artifacts) has taken root in a variety of disciplinary approaches to the study of technology, society, and humanity. (Winner 1986; Latour 1992, Hughes 2004; MacKenzie and Wajcman 1985) A pragmatic turn from this largely descriptive posture sets forth values as a design aspiration, exhorting designers and producers to include values in the set of criteria by which the excellence of technologies is judged. For those who commit to the goal of creating systems embodied with values, the ideal world is one whose technologies further not only instrumental values such as functional efficiency, safety, reliability, and ease of use, but also the substantive values to which societies and their peoples subscribe (E.g. Friedman and Nissenbaum 1996, Mitcham 1995, Nissenbaum 1998). In technologically advanced, liberal democracies, such values may include liberty, justice, enlightenment, privacy, security, friendship, comfort, trust, autonomy, and sustenance.

It is one thing to subscribe, generally, to these ideals, even to make a pragmatic commitment to them, but putting them into practice, which can be considered a form of political or moral activism, in the design of technical systems is not straightforward. Experienced designers will recall the not too distant past when interface, usability, and even safety were overlooked features of software system design. While these have edged into the mainstream of research and practice we are still at the shaky beginnings of thinking systematically about the practice of designing with values in mind. Even conscientious designers, by which we mean designers who support the principle of integrating values into systems and devices, will not find it easy to apply standard design methodologies, honed for the purpose of meeting functional requirements, to the unfamiliar turf of values.[2]

At least two sets of factors contribute to the difficulty of taking values into consideration during design. One which strikes us as epistemological in origin stems from the need to incorporate diverse and frequently far-flung areas of knowledge and know-how into the design process that are not normally conceived as an element of the

design toolkit. Design teams that commit to taking values seriously might even need to innovate in these unfamiliar terrains. The other set of factors stem from the scarcity of explicit guidelines, or methodologies, for reliably embodying values in software systems, in particular, in other technologies, more generally. We refer to these as practical. Accordingly, this paper divided into two parts. In Part One, we describe the epistemological challenges facing conscientious designers. In Part Two, we address the practical challenges by offering our own methodological framework for systematically bringing values, into the process of design, applying it to RAPUNSEL, a research project aimed at designing and building a game environment for teaching girls to program computers (http://www.RAPUNSEL.org).

Because we view these epistemological and practical factors as two dimensions of a single phenomenon, with neither taking priority, the order in which we have chosen to discuss them is somewhat arbitrary. Readers with primarily pragmatic interests can skip ahead to Part Two, dipping back to Part One as needed. Those less concerned with process and method may skip ahead to the Conclusion after completing Part One.

Part One: Epistemology

One reason the study of human and social dimensions of technology is so demanding is that the areas of knowledge and the methodologies it straddles are traditionally far-flung and self-contained. This separation is reflected in the disciplinary organization of almost all universities where the study of technology itself, through the basic sciences and engineering, is typically segregated from the study of social science and humanities. When undertaking the practical task of developing technologies with attention to values, however, designers must engage simultaneously with these distinct areas of knowledge and their respective methodologies. For this task, their intellectual distinctiveness and historical separation is problematic.

Normally, designers of technical systems and devices hold some variables fixed, as background assumptions, while they model design alternatives in search of ones that offer the most efficient means to achieve particular functional ends. When values are introduced as ends, designers must grapple with this extended set of variables, but more importantly, variables that lie outside their usual fields of expertise. Specifically, this means that design and engineering projects must incorporate contextual knowledge about values and, where such knowledge is not readily available, designers will need to

grapple directly with questions about the relevant values. Not only does this lie outside the usual boundaries of engineering expertise -- whether theory or practice -- but is attainable through modes of inquiry unfamiliar in the technical and scientific environments.

Achieving technical design that soundly incorporates values requires not only competence in the technical arts and sciences, but also a reflective understanding of the relevant values and how these values function in the lives of people and possibly groups affected by the systems in question. Within the academy, systematic reflection on values generally takes place in humanistic areas, such as moral and political philosophy, as well as in empirical and theoretical social sciences. For some of the purposes of technical design, what is readily drawn from these fields is sufficient, but for others, the puzzles raised by technologies push beyond standard boundaries. Both circumstances demand more comprehensive interactions among diverse areas of knowledge than is the norm; in the first instance calling for sufficient familiarity to locate relevant insights from prior work, in the second, requiring deliberate efforts to extend what is known in order to address the hard and sometimes novel questions technology raises. The metaphor of “balls in play” captures the nature of these active interdependencies because conscientious designers must juggle and keep in play relevant dimensions of at least three modes of inquiry; foremost they must engage actively with scientific and technical results. Beyond these, they will need to absorb relevant philosophical reflections on values as well as results of empirical investigations of values in relation to individuals and their societies. With the metaphor of balls in play metaphor, we call to mind the need to maintain attention on all three dynamically evolving dimensions simultaneously, not only keeping an eye on each individually, but also on the ways they interact with, or move in relation to one another.

A. Technical Mode

In the technical mode, a designer or design team brings to bear state-of-the-art knowledge on design specifications that might realize given values within the context of an overarching design project. In a project to build a hospital’s patient record system, for example, they may be charged with the task of building privacy protection into the system. Responding to this charge, they might conclude that privacy is best approximated by a system that offers variable access to fields of information depending

on the status of members of the hospital staff. With this in mind, they set about designing the system, identifying the necessary mechanisms to achieve these specifications. In some cases, they may draw on existing state-of-the-art. Where nothing suitable can be found they may need to invent, to innovate, or to settle for a compromise. These iterative steps toward the design and development of a system will not be unfamiliar ones for systems designers; the key departure is the focus on embodying values in addition to conventional requirements of functionality and efficiency.

B. Philosophical mode

Whereas the technical mode calls on designers and engineers to seek and invent mechanisms, the key demands of the philosophical mode are to address questions about the origin and scope of relevant values, their conceptual meaning, and the basis of their prescriptive force. Although the deep and contentious question about the nature of values, in general, is foundational to the other three, it cannot be addressed in the context of this work. Too little has been written about it within the domain of analytic philosophy; here we simply assert a commitment to a general construction of values as purposes, ends, or goals of human action and our intention to focus on those values that are generally construed as social, moral, and political, including in this wide-ranging category abstract values such as freedom, autonomy, equality, justice, equality, democracy and privacy, and more concrete values such as friendship, safety, sociality, and comfort.

Questions about the origin and source of values are of particular interest to creators of technologies if they consider who might use or be affected by systems under construction. They may worry whether values in question are universal to all humans or pertinent only to local groupings only, whether nations, societies, cultures, religions, communities, or families. They may wonder how one best answers such questions, whether through philosophical analysis alone or through empirical methods of the social sciences. Although the most general forms of these questions lie outside the scope of this discussion, they ought to be asked in concrete form, when engineers and designers settle the parameters of their systems and determine the people and groups likely to be affected by them. Although designers and developers of technology in the United States (and other technology producing liberal democracies) may confidently assume the importance of constitutional values of free speech, association, and, and equality, due

process, privacy and property, or others values such individualism, autonomy creativity, and liberation, they should, at least, give consideration to the value commitments of other countries where these technologies are likely to be distributed.

Here, too, the study of values in technologies pushes us to ask questions about origins and sources that are far from settled in the general study of moral and political values. In the absence of clear answers to these questions, recourse to a pragmatic middle-ground seems reasonable. This means going along with the idea of a hierarchy of values in which a “thin” set are taken as common to all humanity and thicker sets associated with groups and subgroups of varying sizes and compositions (e.g. Moor 1999). How this bears on the responsibilities of designers and engineers who, increasingly, are developing technologies for global use, is another question of great interest. On the question whether values can be derived, analytically, or must be discovered, empirically, a middle-ground sees virtue in both, drawing conceptual clarity and normative justification from theoretical works in moral and political philosophy, but supplementing these with knowledge about actual interpretations and value commitments of populations relevant to technologies under study.

In addition to these general questions about values that might be raised in relation to technical design and development, the conceptual investigation of particular values is of great importance for the integrity and soundness of a given project. In undertaking to promote values, such as autonomy, equity, privacy, or well-being, through technical systems, the choices designers make in shaping these systems will be guided by their understandings of the value concepts. In the case of the electronic patient record system, it is important that designers have a sound conceptual grasp of privacy so that their specifications define a system that yields privacy, and not something else. A critic of the proposal, mentioned earlier, to offer variable access to the different fields of information according to the status of members of hospital staff, might say that this design does not properly model privacy. A version whose default is to give access to the patient only and to others only if the patient allows it is better. The point of the example is not to settle the question of which design is preferable but to show how different understandings of privacy, the first case, as contextual integrity (Nissenbaum 2004), for the critic, as control over information about oneself, may be material to the nature of design.

A sound grasp of value terms is one of the necessary links between values and specific design features. Because of this, a significant part of keeping the philosophical

ball in play is to develop concrete analyses of value terms for application in the design context. In the case of technologies developed within industrialized liberal democracies, centuries' long traditions of secular philosophical and political thought is a good, if daunting, source of insights, particularly for some of the contested and difficult value concepts that may emerge. Failure to take these concepts seriously can lead to bungled interpretations in the specification of design features. Tradition, however, may not be sufficient, because technology, a source of radical change in the social and material world, sometimes forces a reconsideration of value concepts. Information technology, for example, has stimulated such reconsideration of the meaning and value of privacy. Keeping the philosophical ball-in-play means remaining astute to these demands and, where necessary, generating an analysis at the appropriate level of concreteness, may call for original philosophical research on the analysis on the affected concepts. One hopes that, over time, a library of analyses would be developed in service of the technical design context. Ideally, this would relieve the burden on designers, allowing them to draw on concrete definitions of past work rather than having to grapple directly with abstract conceptions of key values.

Finally, the philosophical mode of inquiry contributes to the effort of embodying values in design by articulating the rationale behind, or the justification for, commitments to particular values in a given system. Here, too, traditional moral and political theories are a source of explanation for why and when certain values ought to be promoted. This is particularly important when conflicts among values result from specific design choices and background theories may guide sound resolutions, or reasonable tradeoffs. With the electronic patient record system, for example, designers might need to draw on underlying theory to persuade hospital administrators not only that privacy is relevant, important, or necessary, but that it should be protected even if the cost rises, or access is slowed, as a result.

C. Empirical mode

Empirical investigation answers questions that are as important to the goal of embodying values in design as the philosophical and technical. Not only does is it complement philosophical inquiry into what values are relevant to a given project, but it is the primary means for addressing, systematically, the question whether a given

attempt at embodying values is successful, that is, whether designers achieved their intended goals.

As noted earlier, philosophical inquiry provides only one route to determining the values that ought to be considered. Even if we allow for the existence of a basic set of universal human values, many projects will take into consideration a far larger set of values, which may be determined by the cultural, historical, national, ethnic, and religious affiliations of those affected by them. It may be even more crucial to attend to these commitments when designers face a choice among design alternatives that favor some values over others because these could result in significant biases in favor of some users over others. Moreover, despite the enormous attention philosophers, and others, have paid to the problem of resolving values (and rights) in conflict, it remains notoriously difficult (e.g. Richardson 1994). Where practice requires decision, even in the absence of philosophical resolution, a sound alternative is to turn to empirical investigation of relevant populations, ascertaining their commitments and preferences through such mechanisms as surveys, interviews, testing under controlled conditions, and observations in the field.

Another task for which empirical inquiry is essential is in ascertaining whether a particular design embodies intended values. In the case of the electronic patient records system, for example, designers might be satisfied with the efficacy and reliability of particular security mechanisms for providing variable access but be disappointed to find, upon studying usage patterns, that users find them too onerous, most simply bypass them, leaving the records more vulnerable than ever. Through empirical inquiry, therefore, they discover that their attempts to promote privacy are thwarted by a design that does not achieve the intended results. Here, too, various empirical methods may be useful, including usability testing, field studies, surveys, interviews, and so on.

D. Balls in Play

It is not only that the systematic consideration of values in design relies on results from three modes of inquiry. The metaphor of balls in play adds to the picture the dynamic nature of their respective contributions. Within a single project, findings from one area shape what happens in the others, which, in turn, feed back in the first. Because progress is iterative in this way, design team members cannot seek solutions in each of the areas independently of the others but must keep an eye on the other two

even while focusing on one. In the toughest design projects, calling for innovation in all three modes, experts will need to coordinate their efforts, progressing iteratively through the various stages. In less demanding cases – probably more common – designers will be able to rely on what is already known in at least one or two of the modes. As an example of the latter, consider what is needed to build an information system that embodies distributive justice. Designers might, without too much trouble settle the philosophical question of a reasonable concrete interpretation of the value, namely, robust access to people with a diverse range of mental and physical capacities but quickly settle on accessibility to all mentally able individuals as the embodiment of the value of fairness while struggling with the technical questions of how to go about doing so and, later, testing empirically whether their designs have succeeded. Over time, with sustained attention to the study of values in technology, we would expect a body of findings, experience, results, definitions, etc. will gradually alleviate some of the epistemological burdens.

Part Two: Practice

The practical challenge conscientious designers face, is due, in large part, to the sparseness of methodologies and, relatedly, the newness of the endeavor. If we characterize what designers need to know, metaphorically, as the ingredients of a recipe, the challenge of practice is coming up with a method for reliably combining these ingredients into a dish. The aim of the rest of this paper is to sketch one such methodological framework for incorporating values during design, demonstrating its application on a research project in progress, RAPUNSEL, a multiplayer game environment to teach middle-school aged girls to program in Java.

A. The Design Context

In putting forward a methodology for incorporating values in the design process, our goal is not to replace, in blanket fashion, well-established, general design methodologies (Norman and Draper 1986), or the iterative design methods specific to game design (Crawford 1982, Zimmerman 2003), but rather to supplement them.[3] Our stance places values among, or alongside, other criteria of excellence in technical design, such as functional efficiency, reliability, robustness, elegance, and safety, and recently, usability,

and our methodology will augment other approaches which target these criteria. Ideally, the reciprocal consideration of functional factors together with values will result in designs that are good from material as well as moral perspectives. While, here, we focus on the goal of designing software systems that manifest important social values, our background assumption is that successful systems will also meet traditional design criteria. We do not undertake the job of integrating the relevant methodologies, but ideally such a hybrid approach will evolve.

The framework we have developed for incorporating values in software design owes a debt to other important, related efforts. Participatory Design, for example, developed in the Scandinavian context to address concerns over new technologies that have entered the workplace, has built a substantial body of theory as well as a record of application. Its central tenet is that democratic participation by workers in the design of workplace systems is essential not only because it is likely to produce workplace technologies that enhance efficiency and quality of product but also as the skill and well-being of workers themselves. Value Sensitive Design is another approach to incorporating values in design that gives systematic consideration to the interests of all stakeholders in relation to a system under consideration. It shares with the framework discussed below, the crucial relevance of empirical, philosophical, and technical modes of inquiry to the sound inclusion of values in design. (Friedman 1996) Finally, various aspects of our framework have been influenced by Reflective Practice, an approach advocated by practitioners such as (Schön 1983) and Critical Technical Practice advanced primarily by computer science practitioners in artificial intelligence (Agre 1997; Dourish, Finlay, Sengers and Wright 2004; Mateas 2000).

Design, in general, is one among many steps that determines the shape of the technologies that fill our world. Other key junctures are foundational discoveries in science and engineering,, production, testing, marketing, distribution, and adoption. (Hughes 2004;, MacKenzie and Wajcman 1985; Kline and Pinch 1996). These, in turn, rest on a variety of political, economic, and social interventions, too numerous to mention here. Moreover, the particular steps and particular components vary according to particulars of the technology in question, from huge a sprawling utility systems (e.g. Hughes 2004) to stand-alone word-processing applications. Whether devices and systems embody a given set of values is a function of wide-ranging factors, processes, and contingencies, influenced by economic and commercial dimensions, driven by corporate and marketing decision-making, consumer moods and preferences, and

political, policy, and cultural contexts. Yet, as the juncture in the lifecycle of technologies when ideas, goals, and intentions find concrete interpretation in specifications and prototypes, and when developers see the possibilities as well as the limits of these ideas, design is also a critical juncture for envisioning the values which a system in question embodies. It is primarily for this juncture that our methodology has been constructed

B. Case Study: RAPUNSEL

Our methodology for incorporating values in technology has been influenced and refined by the experience of two of the co-authors, Flanagan and Howe, in their work on RAPUNSEL, a multiplayer online gaming environment. Funded as a research project by the National Science foundation, its central goal is to produce a game that would constitute an attractive medium to teach programming to middle-school girls, particularly those from disadvantaged homes (http://www.RAPUNSEL.org). In the current iteration of the game, each player logs onto and is assigned a home environment which houses several characters. Players attempt to 'teach' these characters to move and behave in a variety of ways by programming them in a simplified variant of the Java language. In one scenario, tying mastery of Java with game performance, players must program characters to perform increasingly complex dance behaviors which, according to the game's narrative, increases the characters' degree of satisfaction across a range of metrics, such as being allowed by a fearsome bouncer into a club, or excelling in a dance competition.

Although game design has formed the test bed for our idea, our intention is to draft a methodology that will extend beyond games into other areas of software design including algorithms, system architecture, code, protocols, and interface. Indeed, for reasons we discuss later, these aspects of software systems may well involve fewer complications than games.

RAPUNSEL is a large multi-disciplinary collaboration aimed at designing and implementing an experimental game prototype intended to promote interest and competence in computer programming in middle-school aged girls. This ongoing, three-year project includes a variety of interlinked components: engineering, pedagogy, interface, graphics, networking and more. These components map roughly to core expertise of the three project Principal Investigators (PIs): coding tasks primarily

managed by the computer science team led by Ken Perlin (New York University); game design led by Mary Flanagan (Hunter College), a new media designer; and educational assessment led by Andrea Hollingshead (University of Illinois). One of the project's outcomes, the game itself, will be distributed online. Beyond this, the larger research effort focuses on other contributions to the field including an innovative approach to game design, appropriately "scaffolded" learning outcomes, and the use of repeatable processes.

In addition to the PIs, the research team includes graduate and undergraduate students in computer science, media studies and other fields who contribute both in design and implementation areas. Further, teachers, parents, scholars and game designers, experts from industry, and children have observed the process and contributed from their specialty interest areas on an ad-hoc basis. A subset of this group, known as "design partners," are middle school girls who advise the project team through formal and informal feedback about elements of the game (Druin 1999). The team works together in face-to-face meetings as well as through an online team WIKI, presenting prototypes, reviewing news and related research, and discussing issues of design, implementation, feedback and assessment.

RAPUNSEL is an ideal test case because it is saturated with values questions and commitments. The project's core hypothesis is that the marked absence of women in the field of technology development, particularly in computer programming, is due, at least in part, the style in which basic scientific subjects are taught. (Unambiguous data on the dearth of women in science and technology is well-documented as, for example, in Brunner 1997; Flanagan 2003, Inkpen 1995, Von Prummer 1994). Accordingly, they proposed as an initial strategy that, for female adolescents, socially-oriented environments might be more conducive to learning mathematics and science. As computer games, especially the online variety, are a significant pastime for the target audience, a high quality, socially-oriented game space in which programming is an essential skill for navigation, interaction, and advancement, emerged as a robust initial project goal (Kafai 1998). This activist agenda immediately placed RAPUNSEL in a politically-charged context (AAUW 2000; Mubireek 2003).

As project collaborators hope to address the needs of players generally overlooked by software industry (and even academic) designers, they explore questions like: What are the kinds of social situations and game goals that can be effectively incorporated into multiplayer game design specifically for girls? What kinds of values do

these decisions bring to the design of software usually considered “activist” by mainstream designers? In attempting to answer these questions, investigators found that in addition to what had been articulated (e.g. in the funding proposal) as primary project objectives, several other core goals emerged. These included authorship, collaboration and creativity, altering status-quo perceptions and biases concerning gender, and enacting new possibilities, overall, for game reward systems, which would apply to a variety of types of learners. These goals were understood, progressively, to instantiate higher order values such as equity, autonomy, and liberation, and raised challenging questions about how to recognize and integrate values as they emerge in practice.

The team encountered other values related questions when deciding how to situate their project in relation to currently popular computer games, which are generally competitive and contain problematic representations of gender and race (over-sexualized characters, dark monsters vs. white heroes), social and hierarchical markings (advanced players with wealth and power), and interaction styles (killing vs. protecting vs. collaborating). The multi-disciplinary research team, in conversation with other design participants representing a diverse range of sectors and interests, generated much discussion over values. Furthermore, as a social system where users engage in frequent interactions and exchanges, RAPUNSEL naturally raises considerations about how software design leads to the engineering of social relations, including right and wrong behavior in the treatment of others. The RAPUNSEL project prototype is still evolving as a test bed for how to embody values in a complex, real-world system. Throughout the paper we pair theoretical propositions with examples encountered in the midst of work on RAPUNSEL.

Finally, it is important to acknowledge that a design project like this, pursued within an academic context, does not address all factors that, for example, a commercial project would be obliged to address. Assured funding from a government granting agency and placement within a research setting invokes a different set of demands from games-design projects in the for-profit commercial marketplace facing uncertainties in funding and potential clientele as well as stiff competition. Our ambition, nevertheless, is not merely to succeed in this particular instance, or this particular (academic) environment, but to develop and articulate methodological principles that can usefully be applied to a variety of cases and segments of society.

C. Constitutive Activities

Three types of activities constitute the method we have developed for systematically incorporating values in the design process: discovery, translation, and verification. These activities, as noted previously, are intended as supplements to and not substitutes for other design methods that may guide a given project. As with the three modes of inquiry discussed in Part One, the constitutive activities should be understood not as steps that designers would follow in serial order, but as aspects, or dimensions, of a single undertaking feeding back into one another, iteratively.

C.1 Discovery

This goal of this activity is to “discover” the values that are relevant to, inspire, or inform a given design project, resulting in a list of values, and bringing into focus what often is implicit in a design project. The specific list of values will vary radically from project to project but here, because our interest is not so much in the content of specific lists but primarily with methodology, we have thought about process; that is, the systematic steps conscientious designer might follow in order to “discover” the list of values relevant to any given project. A promising heuristic that emerged in the context of RAPUNSEL was to attempt to answer the question, “what values?” by reflecting on likely sources of values in relation to the system in question. Building on experience of diverse sources of values, including individuals, institutions, societies, religions, cultures, practices, and traditions, we looked to possible counterparts in the RAPUNSEL design context. Although the four discussed below were robust in this context, we would not be surprised to find that not all possibilities have been covered.

a. Values in the Definition of a Project

Sometimes, values are expressed explicitly in the functional definition of a system, though this need not always be so. Whereas with toasters and word processors, for example, we would not expect to find values mentioned in the descriptions of their primary functions, with RAPUNSEL, part of the project’s purpose, articulated in the proposal document, was “to address gender inequities” and meet the technology learning needs of a sector overlooked by the software industry by constructing “a game environment to teach disadvantaged middle-school girls to program computers"

(Flanagan, Hollingshead, Perlin 2003). Thus, gender equity, considered as an instance of social equity is built into the functional definition of the project to be designed. Likewise, as mentioned earlier, systems and devices may be built with the explicit purpose of “preserving privacy” or “enhancing security,” and so forth.

b. Values that emerge in specifying (gritty) instrumental design features

Design is almost always underdetermined by explicit functional requirements (e.g, Whitbeck 1996), leaving to designers and developers numerous open-ended alternatives as they proceed through the process, including some that implicate values. Such values, as they are encountered, we label “collateral” because they are not present in the functional definition of a project, but crop up as a function of the variety of detailed alternatives designers confront and from which they must select.

Reward System

A game’s reward system is a crucial mechanism for expressing the game’s goals and values. In the RAPUNSEL game, designers opted for a reward system that would reinforce larger project goals of cooperation in emerging social behaviors. (Inkpen 1995). In the initial iteration, the team designed a mechanism based upon care-giving or nurturing, which they understood to have been popular with the target audience in similarly structured games. Although it was clear that alternate reward mechanisms could as successfully teach players the relevant programming concepts and skills, designers preferred the version that would appeal to players’ sense of social interaction and achieve higher-level project goals such as cooperation and fair representation. (Kafai 1995; Laurel 2001; Margolis and Fisher 2002, Muller and Kuhn, 1993).

Player Point of View

Another collateral value emerged in RAPUNSEL when designers attempted to determine the player’s point-of-view. When developing the initial specification the two basic options were: a) allow players to manipulate characters from a three-dimensional, top down, "god’s eye view"; or b) generate a subjective perspective by allowing players to ‘become’ the characters as first person avatars and thus “see” only what the character could see. Team members thought that the choice of one or the other could more strongly affect the way players understood themselves in relation to game characters, the game world, and, importantly, how they conceived their own agency. Specifying point of view according to the first option -- allowing players to program the behavior of characters from a god’s eye view – worried team members because it might encourage players to view the characters as a sort of 'slave race'. But in the second, if players were identified in the game not as "god" but as their avatars, it appeared difficult to get "into the brains of characters” in order to program their behaviors. This assessment was drawn from comparisons with other existing game models, which allow players to control their characters directly with a mouse and thus provide a seamless relationship between a player's wishes and their avatar's onscreen goals. Programming the character's actions through script, however, added distance between player and avatar in an unprecedented way.

c. Designers' Values

Generally unstudied and frequently not noticed, the values of members of a design team, even those who have not had a say in top level decisions, often shape a project in significant ways as it moves through the design process. Beliefs and commitments and ethnic, economic, and disciplinary backgrounds, may frame their perspectives, preferences, and design tendencies, resulting, eventually, in features that affect the values embodied in particular systems.

Diversity
An example of a designer-introduced value in the context of RAPUNSEL was 'diversity," which emerged in prototypes exploring other, more technical, issues. Once "discovered" and discussed, it became clear that this value was of importance to several members of the team and was then included, explicitly, in the list of values. To RAPUNSEL team members, diversity meant not only expanding the general activity of programming across boundaries of age, gender, economy, and ethnicity, but also fostering a diverse range of approaches to learning. Understood in this way, diversity links to high-level conceptions of distributive justice, such as Michael Walzer's "complex equality" which portrays a just world as one in which a variety of principles determine the allocation of social goods across a variety of autonomous spheres, respectively, so that people who may not excel in one sphere, may do so in another (Walzer 1994). Although gender equity, arguably a form of diversity, was expressed in the functional definition of the project at its inception, team members became interested in additional dimensions of diversity, such as learning styles and cognitive abilities. These dimensions motivated the construction of multiple game goals rather than just one, allowing players to earn rewards in a variety of ways. So far, these include points for decorating characters and home-space environments in unique and creative ways, and for sharing code with other players. Another mechanism that, so far, has only been sketched, will enable voting by other players on saved dance routines; positive community evaluation will translate to points in the reward scheme. This, further, will cause the most popular routines to bubble to the top, rewarding players as well as serving to showcase what is possible in the game-world. Other modes of reward include the traditional one of moving to higher levels when a player has mastered a lower one and, for players preferring a competitive style, the option of player-versus-player dance competitions. The idea is that these multiple dimensions of reward strategies will increase the range of players able to enjoy and excel in the game.

5. User Values

Another obvious source of values are users, or potential users of any given system. Having recognized the importance of this key population in determining the values to be embodied in a system, the challenge is how to go about discovering the values that are important to them in relation to a given system. In recent years, as user testing, in the broad sense, has become an increasingly important aspect of systems design,

designers have struggled for ways to assess, accurately, their propensities and preferences. As discussed in Part One, in relation to values in technology, it is important for designers to assess not only what dimensions people care about, but how they rank these dimensions in the event of conflicts that arise in specific instances. In developing an email system, for example, designers might puzzle over whether to protect senders' anonymity, and how important such a feature is -- the capacity to speak freely without fear of retribution -- in comparison with the opposite – identifiability and the capacity to hold senders accountable for what they say. While it is less important which option is preferred, it is essential to obtain accurate measurements on such dimensions, which may often be unclear. Conducting assessments calls for subtle appreciation of wide-ranging factors that play into people's responses to technologies shaped not only by explicit systems of ethical commitments but also by cultural meanings and past experiences with existing, salient products, which may generate a set of expectations that carry over from one context to another. (Studies have shown, for example, that users have expectations about where information will appear on subsequent screens, moving their eyes, in anticipation, before the screen appears (Hornoff and Halverson 2003). Ideally, designers will apply a variety of methods to answering questions about users' values. Results from focus groups, for example, offer one perspective on explicit value commitments but are not always consistent with the behavioral observations of usability testing (Eysenbach and Kohler 2002).

For RAPUNSEL, the team found prototyping to be an essential component in discovering users' beliefs, preferences, and values. They devised and used a variety of prototyping methods, ranging from focus-groups and one-on-on sessions with design partners, to web-based surveys, paper-prototyping, digital mock-ups, to more traditional approaches using test modules implemented in the software. Noting the pleasure users derived from building and dressing up characters and from manipulating them in the game to engage in relationships with other characters via flirting, dancing and other social behaviors, RAPUNSEL designers inferred users' valuation of creative self-expression, authorship, community, and collaboration. One unexpected finding was the attractiveness, to some users, of subversion, which, here, we have interpreted as a dimension of autonomy. (It turns out that playing games such as *The Sims* "wrongly" is a significant pastime for several design partners, as well as many other players of that game.)

Subversion
Although there is potential to use RAPUNSEL in classroom-based programming curricula, it is primarily intended for more informal, entertainment-oriented environments. To succeed in this arena the game needs to compete for its potential user-base with popular computer game titles. As noted earlier, regular meetings with design partner groups revealed a fascination with playing against the rules and engaging in morbid or macabre behavior. Designers labeled this set of themes, 'subversion'. Some of the girls who enjoyed playing popular games such as 'The Sims', often did so in significantly different ways from those advertised, or normally expected. They found ways around programmed "intentions" of the system, spending more time, for example, decorating their virtual houses, making clothing for or even abusing their characters than playing "by the book," and striving to achieve goals set out in game instructions and marketing, such as earning a living or getting promoted at a job. The same principle applied in the context of other games, such as 'Grand Theft Auto: Vice City', in which some design partners expressed a preference for simply driving around a virtual Miami rather than engaging in the primary game narrative. In allowing for such flavors of alternate play, these complex game systems revealed subversion as an important emergent value. Following this theme through in testing with RAPUNSEL prototypes, 11-13 year old players often requested unusual abilities in relation to their characters, such as "Can you make it run off and die?" We interpret this notion of subversion as a dimension of freedom, autonomy, and creativity through resistance to pigeonholing, unnecessary constraint, and a desire for a more active authorial voice in the game context.

Smart code editor
A second case of user-introduced values affected development of the game's "smart code editor" interface. To begin, designers noted early on in discussions that design partners valued transparency (the openness and accessibility of a given system) and self-expression. One design partner, age 12, reflected the interests of many in the user group by asking, "Could we design the setting… and like… design the characters ourselves?" To encourage and maintain such creative interest it was important to facilitate programming in a manner that was easily comprehensible and empowering. Without sacrificing traditional criteria for successful design along the way (efficiency, robustness, usability, etc.), designers worked on inscribing the value of transparency into the interface of the smart code editor, producing a tool that, via coding, gave players access to the majority of the system and allowed them to influence the entire game-world.

Biophilia
We refer to a third example of values discovered via interaction with users as 'biophilia," the tendency for players to be attracted to and care about life-like characters and processes (Wilson 1986), as exhibited in several products popular with the demographic (e.g. 'Tamagotchi' toys and 'The Sims' characters). While many of the design partners enjoyed engaging with human characters, and found babies, birds, and other animals similarly compelling, they were less attracted to characters that were too abstract (geometric shapes, stick figures, etc.). When faced with simple characters made from geometric shapes, girls noted the characters appeared quick and agile, but less than intelligent: "My character is a little… not smart… maybe it could learn to swim?"

C.2 Translation

Where discovery consists of the activity of identifying the values pertinent to a given design project, translation is the activity of embodying or expressing these values in system design. Translation is further divided into operationalization, which involves defining, or articulating values in concrete terms, and implementation, which involves specifying corresponding design features.

a. Operationalization

The values that are "discovered" in the context of a given project may be of a variety of types and take a variety of forms. For values to be accessible to the design context, particularly those that are abstractly conceived, they must be rendered in sufficiently specific and concrete terms. The activity of analyzing and defining value concepts, is what we call "operationalization." Although it relies predominantly on the philosophical mode of inquiry, discussed in Part One of the paper, it may also draw on results from other modes, particularly the empirical. Of course, no matter how well value concepts are operationalized – that is, made concrete -- the efforts of conscientious designers will be seriously undermined if the substantive nature of the value is incorrectly interpreted. With controversial values like privacy, for example, clarity, good intentions, and technical competence, can be misdirected if not coupled with an accurate, reflective understanding of the nature of privacy itself.

Social Justice

As suggested earlier, the persuasive force of gender equity, expressed in the functional definition of RAPUNSEL, is derived from the commitment of liberal democracies to social justice. In the context of RAPUNSEL, social justice has been operationalized as improved mastery of a high status skill, in the target population. Drawing a legitimate connection between the value and concrete mastery of a specific skill, is part of the task of operationalization. The relevant narrative is mediated by a number of key philosophical and empirical propositions. One is the prominent role of information technologies in Western, technologically literate societies of the late 20th and early 21st centuries. Another is the importance of proficiency in mathematics, science, and programming as a source of social and cultural status as well as access to high paying jobs and generally positive career trajectories. A vision of these linkages is clearly seen in Seymour Papert's assertion: "…programming is the most powerful medium of developing the sophisticated and rigorous thinking needed for mathematics, for grammar, for physics, for statistics, and all the "hard" subjects. In short, I believe more than ever that programming should be a key part of the intellectual development of people growing up." (Papert 2004, 38; also Papert 1993) At the same time, data shows, unequivocally, a radical decline of interest in these subjects in female adolescents and,

later on, a dearth of women in associated careers (Brunner 1997, Catsambis 1994, Chaika 1995, Clewell 2002, Kirkup and Abbot 1997, Haller and Fossum 1998; Norman 1990; Pearl, Pollock, Riskin, Thomas, Wolf and Wu 1990). Accordingly, an effort to contribute toward a reversal of this trend can be understood as a push toward greater equity in access to important social goods -- in this instance social status and lucrative employment. The link between access to social goods and social justice has been amply drawn in many works of political philosophy, including well-known contemporary works John Rawls (1971) and Michael Walzer (1994). Finally, the growing body empirical evidence showing overwhelming popularity of networked computer games and social learning environments, such as email, chat, and online gaming environments (Chmielewski 2004, Grinter and Palen 2000) and the success of games used in science education projects (Gorriz and Medina 2000, Inkpen 1995) inspired the creation of the RAPUNSEL . As a medium for raising the appeal of math, science, and programming RAPUNSEL is a tool for promoting social justice – admittedly in a modest way -- by encouraging the development of highly marketable skills.

b. Implementation

The activity of implementation is aimed at transforming value concepts into corresponding design specifications. This occurs, as noted earlier, in tandem with the general design activity of transforming ideas, intentions, and concepts into material form -- in the case of programming, ultimately, into lines of code. Designers working on embodying values in design are simultaneously attending to functional requirements and other constraints. With values, no less than with traditional functional goals, the art of good design involves embodying specified requirements successfully in a given system and may be evaluated by similar methods, such as reflective practice (Schön 1983). As with other constitutive activities, implementation is dynamically interconnected with the others, and can affect and be affected by them in a reflexive manner.

Co-operation through Code Sharing

As mentioned earlier, sharing and cooperation had emerged as important project values and needed to be cleverly implemented in the game. One of the ways designers sought to do so was through development of robust mechanisms for sharing program code among players allowing several participants to work together to achieve goals. To promote core project goals of acquiring and improving programming skills, players were encouraged to write new code, but the systems as designed to make it possible (and easy) for players to share snippets of code with others. After considering various implementation strategies the design team devised a system in which players could compose, accumulate, and transport code segments, through the various stages of the game, in virtual ‘backpacks.’ The backpack serves a similar function to mechanisms in traditional adventure and conflict-oriented games which allow players to gather weapons or armor in a type of "inventory." In addition, an instant message (IM-like) system (known

to be attractive to girls) would facilitate inter-player communication, enabling one player to query another to learn what pieces of code they are carrying, at which point, the other might agree to share some or all of the code segments in their backpack.[4]

Reward system II

As mentioned earlier, the reward system is an important medium for expressing values in a game. Although the mechanisms described above *enable* code-sharing, it is through RAPUNSEL's unique scoring system, which incrementally rewards players both for authoring original sequences or programs and for sharing them, that co-operation is motivated. The reward system monitors backpack exchanges to reward players accordingly. Each time a player's code is viewed by another player, the author receives a few points; when the code is actually borrowed and used by another player, the originator receives many more points, thus encouraging players to concoct the most interesting and inventive dance sequences and agreeing to share them. In sum, through the integration of transportable code with a reward system that encouraged sharing, we were able to implement collaboration in both the technical framework and the game mechanic in an organic fashion. An added appeal of this solution over others we considered was that it rewarded players with the accumulation of knowledge, as represented by code segments, rather than with material items, like weapons, clothing, or money, and thus reinforced other core project values, specifically sustainability and non-violence. Such synergies occurred somewhat frequently in the project; a fact we attributed to our continued attention to ways in which various project elements, whether technical, functional, or values-oriented, might mutually reinforce each other.

Smart code editor II

The first design iteration was a 'type-in' code editor window with 'smart' features (spelling/syntax checking, automatic code-completion, etc.) to assist users new to programming. The hypothesis was that compared with a more constrained interface, the freedom afforded by a type-in method would foster creativity and expressivity. Concurrently, designers wanted to avoid imposing unnecessary layers between code input and resulting character behavior, believing that any such "disconnects" might have negative effects on the efficacy of the system, blurring users' ability to see the big picture as well as complicating users' mapping between input and feedback (Norman 1990). This hypothesis was supported by informal empirical data which suggested that students who had learned to program in various 'drag-and-drop' environments (i.e., Director, Flash, Max/MSP, etc.), found these interfaces placed an unnecessary distance between the user and the internal workings of the system; distance that created confusion and frustrated learning.

Subversion II

To implement the "value" of subversion, the RAPUNSEL game, which is designed in the style of a simulation, is built upon and infrastructure that allows for (and supports) unexpected scenarios and user-interactions. Instead of attempting to imagine and map out responses to all possible game-states, the world is built to run in a consistent fashion with, or without, the usual set of user-initiated actions. In this way, the game supports subversive activity without anyone knowing ahead of time what form the subversion might take, providing the necessary robustness to withstand a wide range of unexpected outcomes. In other words, the basic idea is to build a robust (real-world physical) model

that runs whether or not human players are present, making the characters “smart” enough to deal with unanticipated states by continuing to pursue their goals, without crashing or falling apart. The team also designed an ‘underworld’ and nasty characters called ‘gobblers’ to address user interest in subversion.

c. Values in Conflict: resolving, dissolving, and trading off

Throughout a project, there is potential for incompatibilities to arise in the context of a particular design element. A source of such conflict is when designers who have committed to values that have emerged during discovery, find it impossible to embody all of them within the system in question. Engineering is rife with such conflicts – whether to favor safety over cost, transparency over privacy, aesthetics over functionality, with many more appearing at layers of finer granularity. The case of software is not significantly different. An example that has been much discussed is the conflict between system security and ease-of-use (an aspect of autonomy, or self-determination) where a system that has been designed with a high degree of technical security might require more than an acceptable degree of effort on the part of users. Resolving conflicts of values is by no means a challenge for engineering only, but remains one of the most intractable problems in the field of practical ethics as well as in law, moral philosophy and politics. Although there have been numerous efforts to clarify some of these problems (e.g. Richardson, 1994), those familiar with the field will understand why attempting to offer any simple, across-the-board solutions falls far outside the scope of the paper. Instead, some observations drawn from our experience thinking specific conflicts through, in the context of RAPUNSEL, may be useful for purposes of values in design.

Our experience with RAPUNSEL pointed to two key strategies for dealing with conflicts. In one set of cases, when confronting a conflict of values, designers would discover that the problem was not the result of fundamental incompatibilities between values themselves but the outcome of conflicting material constraints that each of the values seemed to impose on the given system or device. This discovery steered designers to look for solutions in the design itself to see how malleable the material limits are, in some cases, requiring advances in skill or knowledge beyond the existing state-of-art. Cases in which designers are able to find solutions, through creative redesign, which allowed both values to be instantiated, we label, “dissolving conflict.” Where it is not possible to dissolve conflict completely (in some cases, one could alleviate but not

solve) through redesign, the alternative is either to trade one value off against another (or others), per the strength of value commitments, or to seek a compromise among conflicting values.

Player Point of View II

Recall the conflict mentioned earlier between designers' preference for a subjective point-of-view, out of concern that a 'god's-eye' point-of-view might encourage a view of game characters as a "slave race", and the clear preference expressed by most players for a third-person perspective. This conflict, between user-preferences and designer-values, proved to be one that could be 'dissolved' through the integration of a third value on the list, biophila. As discussed earlier, design partners had shown a keen preference for engagement with life-like agents and processes. By leveraging biophilia, specifically by implementing a handful of simple AI techniques that provided a degree of autonomous behavior to non-player characters, designers were able offer a god's eye view that avoided slippage to a conception of characters as mere slaves.

Smart Code Editor II

In our earlier discussion of the 'smart code editor, a programming interface for creating code, we reported that team members initially preferred a type-in version, thinking would be less confusing and constraining than a graphical editor. But they discovered that most of the middle-school design partners disliked typing their own code. Reporting that typing felt more like work than play, they made clear that they did not appreciate its advantages; for example, the sense of empowerment designers had hypothesized. In preferring the ease-of-use and efficiency offered through the direct manipulation of objects, via mouse input, players' values conflicted with core project values of exploration, creativity, and empowerment. The resolution, in this case, was a tradeoff: a tiered interface system that began with a context-sensitive, pull-down menu, transitioned to a hybrid of menu and typing, and finally offered the ability for typing directly in a smart editor window equipped with assistive features such as keyword-completion, syntax-checking, and 'real-time' compilation.

Character Representation II

The appearance, attitudes, and actions of characters have significant expressive meaning. Typically, enemies in popular commercial games are depicted as dark "others", whereas heroes tend to look muscular and are often Caucasian. Characters act or speak in ways that mark them culturally and socially, and these markings are likely to influence the way other images and situations are read (Taylor and Jakobsson 2003). In RAPUNSEL, designers found that translating equity into particular character representations was a task of considerable difficulty. They found that relatively small alterations in a character's looks or behaviors could ripple out to alter meanings in subtle and important ways. For example, whether characters (even represented as abstract shapes) would be viewed as male or female was of concern both to designers -- who were keen to create proactive female, or at least gender-neutral, characters -- and to users. At first, the team used simple animated geometric shapes as characters, but these were often perceived as "male" by our design partners and, moreover, were uninteresting to them. When design partners were asked if they would like to care for one of these abstract characters, a 14 year old replied, "No, they just aren't… cool

enough."

To investigate this trend, the team created an online character survey of user preferences for a variety for character representation styles. The results of this survey showed that design partners consistently preferred overtly sexualized female figures over animals or abstract shapes. For example, one user stated, "I didn't like any of them, I just chose the ones that look normal". The design partners linked their preferences to the existing commercial products they enjoyed. Overall, the favorite character, a hip-hop styled female from a popular girls' website, was regarded by one 11 year-old design partner as a "cool girl… she's modern, art-time; she has attitude." Facing the implications of these findings, several members of the team, nevertheless, wished to resist reinforcing the stereotypical images of women often found in popular games. This conflict is a complex one, pitting ideologically driven system designers, against potential users, as well as background societal norms.

Although these tensions have not, to this point, been resolved; tradeoff and compromise is the most likely path. Designers will not concede to the overly sexualized figures preferred by design partners but may yet opt for gendered characters of a more "sporty" type. Team members continue to study the issue, and a variety of alternatives, through both empirical and philosophical modes of inquiry.

C.3 Verification

In the activity of verification, designers assess to what extent they have successfully implemented target values in a given system. Verifying the inclusion of values is likely to draw on strategies and methods not unlike those applied to other design criteria like functional efficiency and usability. These may include internal testing among the design team, user-testing in controlled environments, formal and informal interviews and surveys, the use of prototypes, traditional quality assurance measures such as automated and regression-oriented testing, and more. Verifying the inclusion of values, however, introduces additional layers of complexity as it is important to determine not only that a particular value has been successfully implemented in a specific component, but also that its implementation does not detracted from other design goals. To further complicate matters, our grasp of what it means for a value to be implemented in a system (e.g. claiming that a system is "privacy-preserving" or "autonomy enhancing"), is not nearly as firm as our grasp of what it means (according to the state of the art) for a system to function properly or be "usable". This difficulty arise not only from the more controversial and less concrete nature of value concepts – compare autonomy to usability – but because the means by which values are embodied are often more diverse. This point will be taken up again in the paper's conclusion where we discuss the difference between values expressed directly in a system's content, and those embodied through material constraints or affordances that systematically prevent or enable certain

types of actions. Although values may be related to specific system features (this is a core assumption of our work), they may also emerge, indirectly, as a property of the system's interaction with the contextual setting in which it operates. A final complexity involves the fact that the impact of some values may be experienced immediately, while others may emerge only in the long term. Although significantly more could be said on this subject, it should be clear that within the active design phase of a project, verification is likely to produce only partial results. Despite this limitation, values verification activities are an essential part of the iterative feedback loop.

In RAPUNSEL, the design team has explored diverse modes of verification, sampling from the range of empirical, technical, and conceptual (or analytical) approaches mentioned above. Testing via prototypes (e.g. Glass 2000, Laurel 2001, Retiig 1994, Zimmerman 2003) was found to be particularly useful in sorting through the complexities of values-oriented tradeoffs. An iterative approach to assessing design outcomes, based upon a combination of methods, has allowed the team to move quickly through design cycles, incorporating feedback from a wide range of collaborators and users into successive prototypes. (Bødker and Grønbaek, 1991; Eysenbach 2002, Shneiderman 2000). In addition, the team has employed agile programming methods to aid in implementing technical aspects of the systems with changing requirements (Freeman-Benson and Borning 2003). In the RAPUNSEL process, verification has been facilitated through regular meetings with design partners, educators, and industry advisors. As promising ideas emerge, designers have tried to map these onto functional, technical, and values-oriented frameworks. At the same time, designers have been careful not to overlook the importance of playability and sheer entertainment-value in order to ensure the successfully of RAPUNSEL as a game. Balancing this range of goals has proven a formidable challenge; one which would only have been more difficult without regular cycles of verification integrated within the process.

Player Point-of-View

Findings so far suggest that introducing "free will" into characters, as described above, has had the intended effect. At times, for example, players have had trouble "catching up" with their characters. One 12 year-old design partner noted that they appeared quick and agile; "they move really fast!". Such agency was apparent to the player when characters enacted surprising behaviors and created interesting tensions as they temporarily 'resisted' player instructions in order to satisfy their own 'personal" goals, pursuing behaviors dictated by their personalities and body types. By programming selected aspects of autonomy into character behaviors, RAPUNSEL designers were able to establish a balance – on the one hand avoiding an overly controlling relationship,

while on the other, still supporting the consistent capacity of players to program their characters.

Character Representation III
As described above, although issues surrounding character design have yet to be resolved, the team has developed several methods for obtaining feedback and verifying potential solutions. The online survey of a wide-range of characters, including images from familiar websites ('Neopets', 'WhatsHerFace.com', etc.) as well as RAPUNSEL's own animal, human, and geometrically-generated characters, has elicited much useful feedback. A configurable mini-game environment that allows users to toggle between a range of character types, has also provided detailed feedback on users' preferences. Eventually these and other measures will be used to verify that characters selected for RAPUNSEL, as well as their modes of interaction, successfully embody project values while still engaging users. It is worth mentioning that developing and applying verification measures before a solution has been found, is analogous to 'test-driven development', a relatively new technique in software engineering that has emerged out of the agile programming community. Although a full discussion of this approach is beyond the scope of this paper, we note that it maps quite well to values-oriented methods (Freeman-Benson and Borning 2003).

Smart Code Editor III
Recall the debate surrounding RAPUNSEL's "smart code editor" in which conflicts arose between ease of use and efficiency, on the one hand, and important project values of creativity, empowerment, and transparency, on the other. As noted, the design team settled on an editor with a scaffolded interface designed to 'grow with users' as they acquired experience, affording increasing dimensions of freedom with each successive transition. Thus, typing becomes available to users only when situations arise which necessitate additional expressivity. Usability tests with design partners have borne out these predictions, motivating eager users to pursue increasingly daunting tasks and gain access to the more powerful tools. One 12 year-old, for example, declared, "I want to knock down buildings… and want to design what I want it to be like."

Summary of Methodology

In Part Two, we outlined a systematic approach to embodying values in design constituted by three distinct activities: 1) Discovery, in which a list of values relevant to a project are compiled; 2) Translation, in which values are operationalized and implemented in material design features corresponding to these values; and 3) Verification, in which implementations are tested using variety of methods in order to ascertain whether designers' intentions have been met. It is worth repeating that these activities are not necessarily pursued in any particular work cycle. Although discovery is likely to take place early on in a project, the process need not be completed until quite a bit later, or, in some cases evolve all the way through until a product is, literally, out the

door. Discovery can persist long into a project not simply because designers have not been sufficiently thorough or astute, but because as long as a design is evolving through iterating phases of translation and verification, and new (gritty) features emerge, there is always the chance that new versions will bring to light new value choices. Likewise, with verification: although it might seem marked as the capstone activity, it, also, works best if performed continuously throughout a project in a dynamic feedback loop with translation and discovery. And, so on.

Conclusion

Specific features of the approach we have presented are likely to evolve upon further testing and critical scrutiny. Challenges that drive closer to the heart of the enterprise, however, are those that question its fundamental premises: first, that technologies do, in fact, embody values, and second, possibly more controversial, that values may be embodied in technical systems by design. The first of these premises positions our work within an extended discussion in the philosophy and social study of technology, placing it at odds first and foremost with claims of technology's neutrality. This premise does not deny that technical artifacts do not have or have not had significant effects on social and political aspects of life, nor does it deny that technologies have affected the preeminence of certain social, moral and political values over others. It denies only that whether or not they do is a function not of system characteristics but of the uses to which they are put; technologies are mere tools of human intention. Morality (good or evil) and politics inhere in people and not in their tools. In contrast, our premise asserts that technical artifacts may bear directly and systematically on the realization, or suppression, of particular configurations of social, ethical, and political values.

Accepting that artifacts embody values leaves open the question of how, or by what means they do. Our work with RAPUNSEL suggests an important dichotomy not fully recognized in the paper's main narrative -- between those values we might say are *expressed* in the game and those which are materially embodied. The former applies more readily to game content, such as to issues of, say, character representation or game plot. In this, RAPUNSEL resembles other creative works, such as works of literature, film, television, and so forth, which express and symbolically represent values but whose systematic effects on readers, audience, etc. and communities, is an issue of

great contention. The latter applies to considerations of the material constraints and affordances that the game generates by means of, for example, the code-backpack, “peep-chat,” and the reward system. In this RAPUNSEL resembles other material artifacts which can be shown to bear deterministic, even if complex, relations to social outcomes. Moreover, although the precise nature and extent of this relationship is contended, the metaphor of embodiment is meant to signal that it inheres not only in a correspondence between values and the character of a system as a whole, but in possible micro-level correspondences between values and specific, or detailed system features, including some hidden from plain sight. The latter is especially pertinent to computer software. Although an elaborate analysis of these issues is beyond the scope of this paper, they are of great relevance to the larger enterprise, and worthy of extensive study and discussion.

As mentioned earlier, our analysis assumes far more than that technical artifacts embody values; it assumes the possibility of embodying values in technical artifacts by design, an assumption that is disputable from a variety of perspectives. For one, those who support the neutrality thesis will find this a non-starter. Supporters of a version of social constructivism, too, are likely to be as skeptical about whether the efforts of conscientious or, for that matter, manipulative designers of might result in the promotion or suppression of particular values. It is not through material design features that artifacts embody values but through the meanings they acquire and interpretations they are given as a function of political, historical, cultural factors, as well as a myriad uncontrollable social contingencies. (E.g Woolgar 1991, Pinch and Bijker 1987, Pfaffenberger 1992) Because these meanings and interpretations are not a determinate product of material design, it is pointless for designers, or other parties to the development and production of technical artifacts, to set about the task of embodying values. A third, far more pragmatic line of reasoning, stemming from general skepticism over the power of designers to shape technology and its outcomes, reaches a similar conclusion. Given the innumerable junctures through which any noteworthy technical project must pass, from its conception to realization in a manufactured product, it is unlikely whether even the best intentioned and best informed designers among the many other agents directing critical decisions points are in a position to determine outcome. This general dilution of agency is further amplified by the notorious unintended consequences often associated with new technologies.

In holding to the “obduracy” of technology, (MacKenzie and Wajcman 1985), we are unconvinced by social constructivist approaches that deny any deterministic link between technology’s material forms and social and political outcomes. Yet these, and the other challenges mentioned above, underscore a need to go beyond naïve deterministic truisms in striving to understand and influence this linkage. Accepting that technologies are capable of entering into deterministic causal relationships with social, political, and moral phenomena, does not mean ignoring complexity. Material features are important but so are others, such as the individual, social, historical, and political contexts within which technologies operate. The relevance of these factors is perhaps, hardest to fathom in the case of expressive values and the nature of the communicative chain from author, artist, designer, and producer, to reader, audience, user, etc., and worried leaders try to predict or forestall the influence of, say violent, sexually explicit, and bigoted content on beliefs and behaviors. Yet there remains a point of irreconcilable difference between the critical views mentioned and our own, which bears most strikingly on where to locate the burden of responsibility for the nature– the harms and benefits – of technical artifacts. Our view keeps designers determinedly in this picture. Obviously, anyone can be political; the question is whether it is in their capacity as designers that they are political. We hold not only that they are, but that it is the duty of good designers to embrace this dimension of their work, even if they are not always able to prevail against the tide of countervailing forces.

6. References

1. Ackerman, M.S. and L. Cranor. 1999. *Privacy critics:* UI components to safeguard users' privacy. *Extended Abstracts of CHI* 1999*:* 258-259.
2. Adler, P.S. and T.A. Winograd 1992. *Usability: Turning Technologies into Tools*. New York: Oxford University Press
3. Agre, P.E. 1997. Introduction. In *Technology and Privacy: The New Landscape*, edited by P.E. Agre and M. Rotenberg, 1-28. Cambridge: MIT Press.
4. Agre, P.E., 1997. Toward a Critical Technical Practice: Lessons Learned in Trying to Reform AI. In *Bridging the Great Divide: Social Science, Technical Systems, and Cooperative Work*, edited by G. Bowker, L. Gasser, L. Star, and B. Turner. Hillsdale: Erlbaum.
5. American Association of University Woman. 2000. "Tech-Savvy: Educating Girls in the New Computer Age", http://www.aauw.org/research/girls_education/techsavvy.cfm.
6. Bødker, S., and K. Grønbæk.1991. Design in action: From prototyping by demonstration to cooperative prototyping. In *Design at Work: Cooperative Design of Computer Systems,* edited by J. Greenbaum & M. Kyng, 197-218. Hillsdale: Erlbaum.
7. Borning, A., B. Friedman, and P. Kahn. 2004. Designing for Human Values in an Urban Simulation System: Value Sensitive Design and Participatory Design. *Proceedings of Eighth Biennial Participatory Design Conference.*
8. Brunner, C. 1997. *Opening technology to girls:* The approach computer-using teachers take may make the difference. *Electronic Learning* 16 (4): 55.
9. Catsambis, S. 1994. *The path to math:* Gender and racial-ethnic differences in mathematics participation from middle to high school. *Sociology of Education* 67 (3): 199-215.
10. Chaika, M. 1995. Ethical Considerations in gender-oriented entertainment technology. Crossroads of the ACM. http://www.acm.org/crossroads/xrds2-2/gender.html.
11. Chmielewski, D.C. 2004. Kids turning to instant messaging. Knight Ridder. http://www.azcentral.com/families/articles/0225faminstantmessage.html.
12. Clewell, B. 2002. Breaking the barriers: The critical middle school years. In *The Jossey-Bass Reader on Gender in Education,* edited by Jossey-Bass, 301-313. San Francisco: Jossey-Bass.
13. Crawford, C. 1982. The Art of Computer Game Design. http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html.
14. Dourish, P., J. Finlay, P. Sengers and P. Wright. 2004. *Reflective HCI:* Towards a Critical Technical Practice. *Proceedings of CHI* 2004.
15. Druin, A. 1999. *Cooperative inquiry:* Developing new technologies for children with children. *Proceedings of CHI* 1999: 592-599*.*
16. Eysenbach, G., and C. Kohler. 2002. *How do consumers search for and appraise health information on the World Wide Web?* Qualitative study using focus groups, usability tests, and in-depth interviews. *BMJ* 324 (7337): 9.
17. Flanagan, M. 2003. Next Level: Women's Digital Activism through Gaming. In *Digital Media Revisited*, edited by A. Morrison, G. Liestøl and T. Rasmussen, 359 - 388. Cambridge: MIT Press.

18. Flanagan, M., Hollingshead, A., and Perlin, K. (2003). HRD-0332898, Gender in Science and Engineering Program (GSE) of the National Science Foundation.

19. Freeman-Benson, B. and A. Borning. 2003. *YP and urban simulation:* Applying an agile programming methodology in a politically tempestuous domain. *Proceedings of CHI* 1999: 80-87.

20. Friedman, B. 1996. Value-sensitive design. *Interactions* 3 (6):17-23.

21. Friedman, B., D.C. Howe and E. Felten. 2002. *Informed consent in the Mozilla browser*: Implementing value-sensitive design. *Proceedings of 35th Annual Hawaii International Conference on System Sciences* 247.

22. Friedman, B. and H. Nissenbaum. 1996. Bias in computer systems. *ACM Transactions on Information Systems* 14 (3): 330 – 347.

23. Glass, R. L. 2000. *Facts and Fallacies of Software Engineering*. Lebanon: Addison-Wesley Professional.

24. Gorriz, C., and C. Medina. 2000. Engaging girls with computers through software games. *Communications of the ACM* 43 (1): 42 - 49.

25. Grinter, R. and L. Palen. 2002. Instant messaging in teen life. *Proceedings of Computer Supported Cooperative Work* 2002: 21 - 30.

26. Haller. S and T. Fossum. 1998. *Retaining Women in CS with Accessible Role Models.* Proceedings of SIGCSE'98 Conference.

27. Hornoff, A.J. and T. Halverson. 2003. Cognitive Strategies and Eye Movements for Searching Hierarchical Computer Displays. *Proceedings of the ACM Computer Human Interaction Conference*.

28. Hughes, T. 2004. *Human-built world: how to think about technology and culture.* Chicago: University of Chicago.

29. Inkpen, K., K.S. Booth, M. Klawe, and R. Upitis. 1995. Playing together beats playing apart, especially for girls. *Proceeding of Computer Support for Collaborative Learning* 1995: 177-181.

30. Kafai, Y.B. 1995. *Minds in Play: Computer Game Design as a Context for Children's Learning.* Hillsdale: Erlbaum.

31. Kafai, Y. B. 1998. Video Game Designs by Girls and Boys: Variability and Consistency of Gender Differences. In *From Barbie to Mortal Kombat: Gender and Computer Games,* edited by J. Cassel and H. Jenkins, 90-114. Cambridge: MIT Press.

32. Kirkup, G. and J. Abbot. 1997. *The Gender Gap.* A Gender Analysis of the 1996 Computing Access Survey. *PLUM Paper 80.*

33. Kline, R. and T. Pinch. 1996. *User as Agents of Technological Change:* The Social Construction of the Automobile in the Rural United States. *Technology and Culture* 37 (4): 763–795.

34. Latour, B. 1992. Where Are the Missing Masses? The Sociology of a Few Mundane Artifacts. In *Shaping Technology/ Building Society*, edited by W. Bijker and J. Law, 225-258. Cambridge: MIT Press.

35. Laurel, B. 2001. *The Utopian Entrepreneur.* Cambridge: MIT Press.

36. MacKenzie D., and J. Wajcman. 1985. *The Social Shaping of Technology.* Milton Keynes: Open University Press.

37.Margolis, J. and A. Fisher. 2002. *Unlocking the clubhouse*: the Carnegie Mellon experience. *ACM SIGCSE Bulletin* 34 (2): 79-83.

38.Mateas, M. 2000. Expressive AI. 2000. *SIGGRAPH Art and Culture Papers: Proceedings of SIGGRAPH 2000.*

39.Mitcham, C. 1995. Ethics Into Design. In *Discovering Desig,* edited by R. Buchanan and V. Margolis, 173-179. Chicago: University of Chicago Press.

40.Moor, J. H. 1999. Just Consequentialism and Computing. *Ethics and Information Technology*, *The U.N. Human Rights Charter* 1:65-69.

41.Mubireek, K.A. 2003. Gender-oriented vs. gender neutral computer games in education. Educational Policy and Leadership, Ohio State University. http://www.ohiolink.edu/etd/view.cgi?osu1056139090.

42.Muller, M.J., and S. Kuhn. 1993. *Special issue on participatory design. Communications of the ACM* 36:6.

43.Nissenbaum, H. 1998. Values in the design of computer systems. *Computers in Society* 38-39.

44.Nissenbaum, H. 2004. Privacy as Contextual Integrity. *Washington Law Review* 79 (1): 119-158.

45.Norman, D. 1990. *The Design of Everyday Things*. New York: Currency/Doubleday.

46.Norman, D. A and S.W. Draper. 1986. *User-Centered System Design: New Perspectives on Human-Computer Interaction.* Hillsdale: Erlbaum.

47.Papert, S. 1993. *The Children's Machine: Rethinking School in the Age of the Computer.* New York: Basic Books.

48.Papert, S. 2005. *The Challenges of IDC*: What have we learned from our past? A Conversation with Seymour Papert, Marvin Minsky, and Alan Kay, *Communications of the ACM* 48 (1) 35-38.

49.Pearl, A., M. Pollock, E. Riskin, B. Thomas, E. Wolf, and A. Wu. 1990. Becoming a Computer Scientist. *Communications of the ACM* 33 (11): 47-57.

50. Pfaffenberger, B. 1992. Technological Dramas. *Science, Technology, & Human Values* 17 (3): 282-312.

51.Pinch, T.J. and W. E. Bijker. 1987. The Social Construction of Facts and Artifacts: Or How the Sociology of Science and the Sociology of Technology Might Benefit Each Other. In *The Social Construction of Technological Systems*, edited by W. E. Bijker, T.J. Pinch, and T. P. Hughes, 17-50. Cambridge: Mass.

52.Rawls, J. 1971. A Theory of Justice. Cambridge: The Belknap Press of Harvard University Press.

53.Rettig, M. 1994. Prototyping for tiny fingers. *Communications of the ACM* 37(4): 21-7.

54.Richardson, H. S. 1994. *Practical Reasoning about Final Ends*. Cambridge: Cambridge University Press.

55.Schön, D. 1983. *The Reflective Practitioner.* New York: Basic Books.

56.Shneiderman, B. 2000. Universal usability. *Communications of the ACM* 43 (3): 84-91.

57.Soloway, E., M. Guzdial and K. Hay. 1996. Learner-centered design. *Interactions of the ACM* 1 (2): 37-48.

58.Taylor, T.L. and M. Jakobsson. 2003. The Sopranos Meets EverQuest: Socialization Processes in Massively Multiuser Games. *Digital Arts & Culture Streaming Wor(l)ds Conference Proceedings*.

59.Von Prummer, C. 1994. Women-friendly perspectives in distance education. *Open Learning* 9 (1): 3-12.

60.Walzer, M. 1984. *Spheres of Justice: A Defense of Pluralism and Equality*. New York: Basic Books.

61.Whitbeck, C. 1996. Ethics as Design: Doing Justice to Moral Problems. *Hastings Center Report* 26 (3): 9-16.

62.Wilson, E. O. 1986. *Biophilia.* Cambridge: Harvard University Press.

63.Winner, Langdon. 1986. *Do Artifacts Have Politics?* In *The whale and the reactor: a search for limits in an age of high technolog*y. 19-39. Chicago: University of Chicago Press.

64.Woolgar, S. 1991. The Turn To Technology in Social Studies of Science. *Science, Technology, & Human Values* 16 (1): 20-50.

65.Zimmerman, Eric. 2003. Play as research: the iterative design process. Game Lab. http://www.gmlb.com/articles/iterativedesign.html.

---

[1] We would like to thank members of the RAPUNSEL team for their dedicated work on the project, especially the Co-PIs Perlin and Hollingshead. Our ideas benefited enormously from audience comments at the Workshop on New Directions in Understanding Ethics and Technology, University of Virginia (particularly Dean Neusma, who commented on the paper), colloquiua in the Schools of Information Science at Bathurst and Wagga Wagga campuses of Charles Sturt University, and at CHI2005. RAPUNSEL is supported by NSF's Program Research on Gender in Science and Engineering, HRD-0332898. Values in Design research was supported by the Ford Foundation's Program in Knowledge, Creativity, and Freedom for the project, *Values in Technology Design: Democracy, Autonomy, and Justice*. We thank Maja Petric for providing excellent research assistance.

[2] Nevertheless, there have been exemplary practical efforts to integrate values into design, such as: for privacy (Ackerman and Cranor, 1999; Agre 1997:; universal usability (Schneiderman 2000); community (Druin, 1999); social justice (Borning, Friedman, and Kahn; 2004), and informed consent (Friedman, Howe and Felten, 2002)

[3] We are grateful to Dean Neusma for suggesting we bridge our work with other more general design methodologies. Although this task is too great for this paper, we acknowledge a need for further development along these lines.

[4] On a technical level, the implementation of sharing and collaboration in the manner described grew complicated as designers considered how players might save and transport pieces of code in various stages of the game. For example, it was clear that code segments for user at different levels should be created and saved at the same level of granularity, both for conceptual clarity and ease of sharing. Yet, requiring the code to conform to a high-level abstraction like a Class or Interface seemed potentially difficult to grasp, perhaps even discouraging sharing among less experienced players. Through several prototype iterations, we addressed this issue by integrating code-sharing directly into the smart code editor in conjunction with a 'name-code' command. This functionality allowed a player to highlight a section of code in their editor and save it as a Java method. When 'name-code' was invoked, the system analyzed the code chunk selected, added a method signature with appropriate arguments, and prompted the user for a memorable name. At

this point, the player could view and/or test the new method and approve its inclusion in their backpack. It was a pleasant surprise when we later saw how this design choice also served an important pedagogical goal, namely that of teaching 'encapsulation'. Our initial belief, that encapsulation (the combination of smaller code chunks into a functionally coherent whole, reusable without reference to its constituent parts) was among our 'advanced topics' and perhaps available only to 'expert' players, was revisited and the concept's importance in the game increased. The iterative dialogue between functional, values-based, and technical concerns again lead designers in interesting new directions, yielding positive externalities well beyond the initial 'problem' considered.