Chapter 7
Predecessor Artifacts: Evolutionary Perspectives on a Reflective Conversation with Design Materials

Anders I. Mørch

Abstract Donald Schön described designing as a reflective conversation with the materials of a situation, and proposed a phenomenological account of designing. Ray McCall proposed an extension and generalization of Schön's model that more easily allows for computer support. Building on the work of Schön and McCall, this chapter adopts an evolutionary perspective on the design process and proposes "predecessor artifacts" as a new concept for understanding the reflective conversation with design materials. Predecessor artifacts emerge in creative activities and they encapsulate design history. The concept is applied to software design. Based on a literature survey, experimental prototyping, and empirical analyses in two user organizations, three types of predecessor artifacts are explored: (1) reusable design material, (2) alternative functionality, and (3) reusable design concepts. A goal with PAs is to understand design reuse in software applications from a phenomenological viewpoint.

Keywords Evolving artifacts • Design theory • Human-centred design • Predecessor artifact • Reflective conversation • Resemblance relation • Reusable design • Software design • User-participation

7.1 Introduction

Exploring predecessor artifacts (PAs) and how they are related to creativity and rationale is the aim of this chapter. Two claims are made: (1) Predecessor artifacts add a historical dimension to the reflective conversation with design materials (Schön 1983, 1992), and (2) connecting an artifact with its predecessor artifacts provides new opportunities for creative interactions with the artifact.

A.I. Mørch (✉)
InterMedia, University of Oslo, Oslo, Norway
e-mail: anders.morch@intermedia.uio.no

J.M. Carroll (ed.), *Creativity and Rationale: Enhancing Human Experience by Design*, Human–Computer Interaction Series,
DOI 10.1007/978-1-4471-4111-2_7,

7.1.1 Predecessor Artifacts and Creativity

Wittgenstein (1967) used the notion of "language game" to describe language use by analogy to how games are played and made sense of. He said that all games have common features, but no one feature is found in all games. Wittgenstein's point was that things, which may be thought to be connected by features shared by all, might in fact be connected by a series of overlapping similarities that are present to various degrees. This form of connecting things, referred to as *family resemblance*, was modeled after how members of a family relate to each other according to traits (visible properties): build, features, color of eyes, gait, temperament, etc. For example, a person may have one or more common features with a relative; they may share strong or weak traits, and the traits may change during the person's lifetime. This reveals a more dynamic relationship structure among things than what is common in software design and object-oriented analysis and design. Schön adopted this language into the domain of architectural design in order to understand the intertwining of drawing and talking, calling it a language for doing architecture (Schön 1983, p. 81), and Pelle Ehn adopted the same language for information systems design with end users, emphasizing the mutual learning of two different language games, one for designers and the other for users (Ehn 1988). The language game metaphor goes beyond formal methods in software by tapping into the richness of human experience and the myriads of ways of connecting reusable things. This is the approach to creativity advocated in this chapter.

7.1.2 Predecessor Artifacts and Rationale

Design rationale has been defined as a method of designing an artifact that makes explicit the reasons for its design (Moran and Carroll 1996). One way to organize design rationale is by distinguishing a process-oriented approach from a product-oriented approach (McCall 2010). The process-oriented approach (Conklin and Burgess-Yakemovic 1996; McCall 1991) is about capturing the argumentation or "design history" behind the artifact (a computer interface, a building, a neighborhood plan, etc.), which is a form of documentation that includes the alternatives considered during the design process, the chosen and the rejected alternatives. The historical sequence of the process is significant. The product-oriented approach is about analyzing the structure and nature of the artifacts (i.e. exploring the space of alternative artifacts) not its process of creation and development (MacLean et al. 1991).

A PA is special form design rationale that combines aspects of the process-oriented and product-oriented approaches. A PA is a rejected alternative (a product) that represents a snapshot in the history of a current artifact (the process). Furthermore, a predecessor artifact is represented by a "link" in a current artifact, which can be a role assigned to it or an emergent property (physical trace, symbolic reminder).

For example, when a new tool incorporate features of one or more existing tools, or when a new innovation makes older tools obsolete, or when a new tool appears on the market even though multiple alternatives are available, predecessor artifacts emerge. The process that leads to a predecessor artifact is a complex one that cannot be predicted in advance and includes creative acts of users, innovative system building, and often involving considerable efforts in engineering, management and marketing. For example, Apple's touch and gesture features of the iPhone with its easy-to-use interface for interacting with software components (apps) is an example of a new tool that has transformed competing tools into predecessor artifacts (i.e. other smartphones that were in the market at the time the iPhone was introduced in 2007 as well as the technology the iPhone is based on). Three kinds of predecessor artifacts are differentiated and discussed in this chapter based on analyses of evolving artifacts in a variety of domains: (1) reusable design material (traces of PAs), (2) alternative functionality (actual PAs), and (3) reusable design concepts (reminders of PAs).

Previous research in reusable design rationale has addressed reuse based on a cognitive framework, e.g. indexing, retrieving, understanding, and modifying prior design knowledge (e.g. Ball et al. 2001). The early stages of the reuse process (indexing and retrieving) appear differently from a phenomenological point of view. The 'here and now' of the present situation (features of a current artifact in use) becomes the center of an evolutionary design process, and reusable design rationales are associated with traces and reminders of previous artifacts in new designs.

The chapter is organized as follows. It starts with identifying some open issues in Donald Schön's theory of reflective conversation with design materials, in particular accounting for evolutionary design and computer support. Three cases in software design provide empirical support for the claims, each highlighting a unique role of predecessor artifacts in evolutionary development. The findings are discussed in terms of creativity and rationale. The chapter ends with suggestions for further work, including a method for deepened analysis of artifacts in design situations and design-based models of human learning and development.

7.2 Designing as a Reflective Conversation with Design Materials

Schön coined the expression "designing as reflective conversation with the materials of a situation" (1983, p. 78). He presented his theory to the design community in the "Reflective Practitioner" (Schön 1983), and later discussed implications for computer support and artificial intelligence in an article based on a conference keynote talk in 1992 (Schön 1992). Drawing on empirical studies in a range of different design situations, Schön profiled his theory as a generic design process that was characterized by complexity, and "because of this complexity, the designers moves tend, happily or unhappily, to produce consequences other than those intended," and the designer "responds to the situation's back-talk" (Schön 1983, p. 79). The complexity

of the situation is partly a result of incomplete information in the beginning of a design process and the multiple interpretations a designer makes of the situation. Additional information appears as "back-talk" or feedback from the design, to which the designer responds by reflection-in-action, modifications, and reinterpretations (Schön 1983). Observation of designers in a variety of situations (professional and novice) and domains (e.g. architecture, town planning, management, psychotherapy) provided the data to support his claims.

There are two aspects of the reflective conversation model that are mentioned by Schön, but not addressed by him specifically: computer support and evolutionary design (Schön 1992). This will be elaborated in the remainder of the chapter.

McCall proposed an adaptation of Schön's model that more easily allows for computer support; combining reflective conversation with design rationale and referred to as *critical conversation* (McCall 2010). McCall and his colleagues developed computer support for critical conversations based on the procedural hierarchy of issues (PHI) method (McCall 1991). It is the historical dimension (process orientation) of design rationale that is added to the reflective conversation model. This implies that a designer considers the alternative issues, positions, and their arguments as they occur during the design process, in effect intertwining two distinct phases of design, *ideation* (the generation of design ideas) and *evaluation* (feedback from the uncompleted design that challenges the designer to devise new ideas). McCall claims that separating ideation from evaluation is one of the shortcomings of the past work on design rationale that goes back to the work of Rittel (1972), McCall's mentor and the inventor of issue-based information systems (IBIS). Schön's theory, however, provides a clue for their integration. By making a simplified model of the theory, McCall and his colleagues proposed to treat *action* as ideation and *reflection* as evaluation. They further claimed that situational back talk based on computerized feedback could trigger creativity when interacting with a computer-based design environment in certain situations (Fischer 1994). This originated with the Janus system, a computer-based design environment for kitchen design that integrates support for action and reflection in the same system (Fischer et al. 1989; McCall et al. 1990), and a series of systems that both preceded and followed Janus, like Mikroplis and Phidias (McCall 2010).

7.3 Evolving Artifacts

Schön's reflective conversation was not about evolutionary design because reusability of design materials was not a central concern for Schön. His approach to design was to explore creative design (combining design materials in new ways) within the constraints of a realistic design space (defined by task requirements, site properties, etc.). Schön explained that the evolutionary perspective of a design is evident in two ways: (1) a designer can evolve a design by appreciating materials in new ways (Schön used bridge design and jazz improvisation as examples), and (2) design intentions evolve during the design process, which for Schön was a source of problem identification (Schön 1992). These forms of evolutionary design correspond to

what this author calls "specific development" or "adaptation," the evolution of an individual artifact (a series of drawings for a new house; the adaptation of a computer application to a user organization, etc.) (Mørch et al. 2009). Schön used the expression "seeing-drawing-seeing" to provide a phenomenological account of this type of evolutionary design (Schön 1992). The work presented here builds on Schön's work by integrating specific development with general development in order to describe predecessor artifacts (PAs) in software design. PAs are concrete artifacts that influence or stimulate a new design and they "reappear" in various forms of expression of the general level information in a new design.

The evolutionary perspective on designing used in this chapter is adopted from the natural sciences as two interdependent processes: (1) the interaction between organism and environment during individual development (adaptation or specific development), and (2) the interaction between specific and general (species) development (generalization and recapitulation). Recapitulation is a mechanism in living organisms for expressing general level information in a new off spring. It provides an analogy for one type of predecessor artifact (physical trace). One example from natural evolution is the gill slits featured in the embryonic development of human infants, serving as the visible remains of a functional organ inherited from a sea-breeding ancestor. The two processes have direct equivalents in the evolution of artifacts: (1) the interaction between a design and its context of use (adaptation or specific development), and (2) the interaction between specific and general development (generalization and predecessor artifacts). In software design, specific development refers to local design and end-user tailoring, e.g. the adaptation of a generic computer application to a user organization, with techniques ranging from workflow modeling to scripting (Mørch 1997). In contrast, general development refers to the work done by professional developers to fix errors and incrementally improve a product at the source code level, and to spawn new products when opportunities arise (Fischer et al. 1994; Mørch et al. 2009). These two processes are often poorly coordinated in software development, resembling the discrepancy of specific and general development in natural evolution.

On the other hand, artifact evolution and natural evolution differ in a number of ways. For example, in the early part of the previous century philosopher Henri Bergson argued that humans co-evolve with their environment in a more constructive manner than natural evolution can explain, and humans can impact evolution by intervention and acts of creative life (Bergson 1911/1993). Human intervention can respond to injustice, seek out areas for improvement, and correct past mistakes. Furthermore, natural organisms evolve by *necessity* (survival and reproduction), whereas artifacts tend to evolve by *convenience* (Basalla 1988), driven by the needs and desires of creative people and the demands of society (simplification, niceties, economic laws like supply and demand, manufacturability, etc.) (Feyerabend and Terpstra 1999; Tomlinson 2008). Therefore, artifacts can evolve along multiple paths that are not constrained by formal relationships; they can be informal as well as formal, which open the way for creative interactions with artifacts, such as appropriation (Carroll 2011), emergence (Seevinck and Edmonds 2008), and perceived resemblance (Mørch 2003) in local design activities. For example Carroll observed

in a Blacksburg community-computing project (a virtual world simulating a real place) that users created their own use of the system in ways that was not part of the system's original design rationale.

Simon introduced the notion of evolving artifact in "Sciences of the Artificial" (Simon 1996). Architectural design and software design are examples of this branch of science. The study of artifact evolution at the specific and general levels belongs in the sciences of the artificial. Furthermore, the author, following Carroll (1994), Fischer (1994) and Norman (2008), takes the everyday uses of artifacts as the starting point for design, e.g., an expansive design process triggered by a difficult to use artifact. Previous studies of evolving artifacts include pottery jugs (Basalla 1988), buildings (Brand 1994), everyday tools (Petroski 1992), and computer applications (Carroll and Rosson 1996; Fischer et al. 1994; Mørch 1996). In these studies, design is characterized by incremental improvements of existing artifacts and to resolve the tensions of adaptation and generalization (Mørch et al. 2009). Traces and reminders of predecessor artifacts emerge as a result of these dynamic processes, in particular as a new design reuses material and ideas from previous designs. For example in the ancient African pottery jugs Basalla (1988, p. 107) analyzed, a cord made of organic material served as the first handling mechanism for carrying them. Later it was replaced by a more durable structure formed in clay and built into the jug itself. The original handles became ornamentation in the later series of jugs, serving as a symbolic reminder of the previous ways of using them (Mørch 2003).

Everyday physical artifacts triggered the author's research interest in predecessor artifacts and how they relate to reflective practice, creativity and rationale. Figure 7.1a shows an example of a reusable design material (parts of an old building reused in a new building). Figure 7.1b shows a reusable design concept, "exterior shelter space" or enclosed porch ("svalgang" in Norwegian), which refers to the lightweight structure built around the building's entrance area to protect people from bad weather upon entering, a concept originating in medieval farmhouses in Norway. Coin-operated and wall-plugged telephones represent a third type of predecessor artifact, serving as backup system to mobile phones.

Schön's conceptual framework for reflective conversation does not have a vocabulary for evolutionary design beyond adaptation. Therefore his framework must be augmented to take into account generalization and predecessor artifacts. This is easier with computer-based design environments than paper and pencil techniques because computational artifacts consist of multiple levels of abstraction (e.g. user interface, program code, runtime organization). However, predecessor artifacts are not exclusively tied to computer applications; they are found in all kinds of physical artifacts as illustrated by the motivating examples.

7.4 Research Questions and Methods

The research presented in this chapter is a reinterpretation of data from previous projects the author has been involved in, now with an eye on describing predecessor artifacts. The first project was part of the author's PhD dissertation (1993–1997),

Fig. 7.1 *Left* (**a**): reusable design material (old timber construction in a new house). *Right* (**b**): A reusable design concept ("svalgang," a kind of porch) refined in Storøya kindergarten (Photos from Bærum, Norway)

and demonstrates how end-user tailoring of a generic application (BasicDraw) can create a domain-oriented application (KitchenDesign), revealing PAs as physical traces of old functionality in the new application. The second project is a case study in the organizational implementation of an integrated work and learning platform for a national chain of gasoline stations in Norway (2001–2004), which resulted in existing tools becoming back-up systems (PAs). The third project is an empirical study of the interaction of end-user development and professional development in a software company that develops project-planning tools for the Nordic oil and gas industry. It was a case in a large European project (2006–2011), and revealed a design concept preserved and refined by the company in all its products. The methods used for data collection include experimental prototyping, interviewing, observation, videotaping, and web survey. The analyses reveal three types of predecessor artifacts: (1) reusable design material (project 1), (2) alternative functionality (project 2), and (3) reusable design concept (project 3).

7.5 Reusable Design Material in Application Evolution

The structure for this and the next two sections is: (1) context of case/site, (2) findings from the study, and (3) analysis of results from the point of view of describing predecessor artifacts and their emergence and role in evolutionary application development.

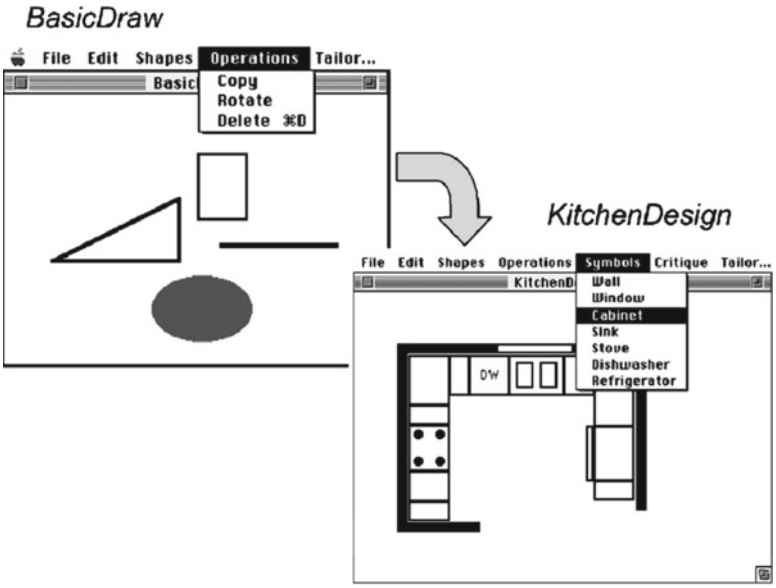

Fig. 7.2 Evolving BasicDraw into KitchenDesign by the tools available in the Tailor menu (customize, integrate, extend). KitchenDesign reuses functionality from BasicDraw, and reveals predecessor artifacts

7.5.1 Context

The goal of this study was to test a hypothesis for the evolutionary development of applications, namely that end-user development tools integrated with a generic application provide the necessary means to evolve the application from one task domain to another (Mørch 1996). Another goal was to bring the inheritance relation between class and subclass in object oriented programming to a more user-oriented level of abstraction (Mørch 1997, 2003).

7.5.2 Findings

We addressed this hypothesis by enabling end users to evolve BasicDraw into KitchenDesign (Mørch 1996). The two screenshots in Fig. 7.2 show before and after stages of this process, which were accomplished with the end-user development (tailor) tools integrated in BasicDraw (Mørch 1997). When using these tools, an end-user developer can modify reusable software functionality (application units) at three different levels of abstraction (user interface, design rationale, and program code).

The menus and shapes in KitchenDesign are subclasses of the menus and shapes in BasicDraw. For example the Symbols menu is a subclass of the Shapes menu. All the kitchen symbols in the work area are subclasses of the rectangle class, with some symbols composed of additional sub-shapes (rectangle, oval, text). The menu items were created in a similar fashion. The operations on the kitchen symbols (e.g. scaling) were realized as specialized methods to extend and constrain the original methods defined in the super-classes (Mørch 1997). The author conducted a video-recorded usability test of BasicDraw with twelve subjects and asked them to perform end-user development modifications to the application, and found that customization and integration were techniques that users could master without much instruction. The extension (writing program code in method bodies), however, required some knowledge of programming and basic skills in object-oriented programming (Mørch 1996).

7.5.3 Analysis

Even though the inheritance relation connected application units at the program code level, e.g. KitchenCabinet class inherits from Rectangle class; it was the resemblance among the application units visible in the user interface that enabled users to understand what reuse could offer them, i.e. *"kitchen cabinet looks like a rectangle."* The basic shapes became, both literally and in effect, predecessor artifacts in the kitchen design symbols. This means that inheritance is reflected at two levels, user interface and program code. The author proposed two types of resemblance relations to model them: perceived resemblance and self-resemblance (Mørch 2003). The first is "inheritance" as perceived by the user and the second is inheritance manifest in the artifact. Self-resemblance is arguable the relation that most faithfully models an application's connection with reusable software functionality (e.g. enabled by inheritance), but perceived resemblance can also be supported by the computer (reusing objects rather than classes).

7.6 Organizational Implementation of an Information Seeking Terminal

7.6.1 Context

A research team from InterMedia participated in the design and organizational implementation of a new information-seeking terminal for the gasoline division of a large oil company (Mørch et al. 2004a; Mørch and Skaanes 2010). This system was delivered as a web portal and installed in retail stations across the country. Participatory design techniques were used during the planning stages and evolutionary prototyping during the portal development stages (Mørch et al. 2004a).

a

b

INFORMATION-SEEKING METHODS BEFORE THE PORTAL (N=34)

Method	%
Ask a colleague	81
Paper catalogs	58
Staff meetings	50
Call a colleague	38
Post-it notes	19
Call another station	19
Product sheets	19

Fig. 7.3 (**a**) Prototype of information seeking portal, and (**b**) information seeking preferences before the portal was introduced

The system contained information about automobile parts, fluids, and hot food preparation procedures, and was meant to serve as an online help system for everyday work (Fig. 7.3a). The portal adoption process lasted for 14 months and data was collected by interviews and a web survey during that period. The use of the system was not mandated, but the station managers encouraged the attendants to use it. Thirty-four respondents completed the survey. Interviews were conducted selected participants across age groups and management levels (Mørch and Skaanes 2010).

7.6.2 Findings

Before the introduction of the portal, the attendants had to make use of a range of resources for accessing information to support their work. The table in Fig. 7.3b gives an overview of these resources, ranked according to frequency of use. Results from the survey showed that 81% of the respondents reported that asking a colleague was the most useful approach when seeking information. Other frequently used resources of information were paper catalogs (58%) and staff meetings (50%). Paper catalogs included vendor specific product manuals containing automobile parts and assembly instructions (Mørch and Skaanes 2010).

After the portal was introduced, 46% of the respondents said they stopped using one or more of the older methods. One of the respondents said: *"It simplifies work to get rid of all the papers scattered around the cash register and to get all this information in one place"* (Mørch and Skaanes 2010). The remaining 54% of the respondents said they continued to use the older methods despite the availability of the portal, and several employees preferred to use the paper catalogs instead of the computerized display in order to find the required information. As one employee said in an interview: *"I am not very good with computers, most of the time it is much faster to use the paper catalogs"* (Mørch et al. 2004a).

According to several of the attendants, it was important to have alternative means for accomplishing daily work. However, the management plans to terminate the production of those methods that are too costly to produce and those that serve only one function.

7.6.3 Analysis

This research project unraveled a complex system of technology and work practice that was comprised of many different tools and resources for information, each associated with a specific generation of work support: browsing a product catalog, using a map book, contacting colleagues, staff meetings, Post-it Notes attached to a display, and a web portal. The attendants would almost always find a way out of a difficult situation when resorting to one of the alternatives. Some of the alternatives would slow down work; others supported on-the-spot problem solving.

At the current stage of technological fluency in the company, removing the sometimes suboptimal alternatives may complicate recovery from a difficult situation and prevent work completion altogether. Many information-seeking methods in the company currently outperform computer-based information seeking. The older technologies serve as predecessor artifacts, providing alternative means to support everyday work, and the age of the user population is an indicator of what type of technology will be preferred. Over time information browsers will improve in quality, making them more efficient for job specific tasks; but equally important older technologies will be serviced less frequently due the cost of maintaining them and as a result of higher digital competency among the employees.

7.7 Reusable Design Concept in Product Line Development

7.7.1 Context

There is a version incompatibility problem latent in evolutionary application development. A locally adapted version of a system, for example, may be incompatible with a future release as a result of two processes that often are not well coordinated (specific and general development). This issue was addressed by a case study in a small software house (company) to investigate the coordination of specific and general software development (Andersen and Mørch 2009; Mørch et al. 2009). The company develops and sells project-planning tools for the oil and gas industry and provides consultancy services, training, and support for these tools. The company has close relationships with its customers. The researchers from InterMedia were invited into the company to give advice on their knowledge management practices for customer relations. The empirical material consists of interview data and a video recorded meeting with key stakeholders (developers and users).

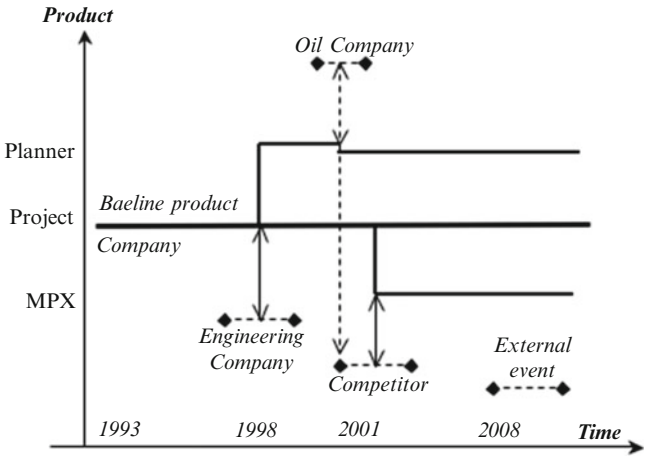

Fig. 7.4 Evolution of a project planing tool (Project) into two specialized products (Planner, MPX) has preserved and refined a design concept that originated in the first (baseline) product

7.7.2 Findings

The company's customers are requested and encouraged to report usability problems and innovative uses; and some of the most skilled users also assist in end-user development (adaptation) of the company's products (Andersen and Mørch 2009). The developers offer communication and information sharing tools for customer interaction, which has been stimulated through long-term relationships (maintenance contracts) and user forums. The main meeting ground is an annual showcase in which customers are invited to communicate with the company's employees.

Understanding the transitions from specific to general development has been one of our research questions (Mørch et al. 2009). The respondents related historical events to some external organizations the company does business with, in particular an oil company and an engineering company. Some of these events led to major changes in the company's product line, including the creation of two new products, Planner and Microsoft Project Extension (MPX). However, most of the external events led to minor changes, producing only gradual improvements and continuation of existing products. Figure 7.4 shows a schematic overview of the factors that have influenced product line development in the company.

A situation where an adaptation for a specific customer led the company to spawn a new product is illustrated in this first excerpt, which was extracted from an interview with one of the company's developers. The developer's view shows that there is a connection between specific and general development, realized when contributions from one customer are incorporated in the product.

Interviewer: Was it an add-on made specifically for Engineering Company?
Respondent: No, it became part of the product. Yes, it started as a patch, what we call a user option.

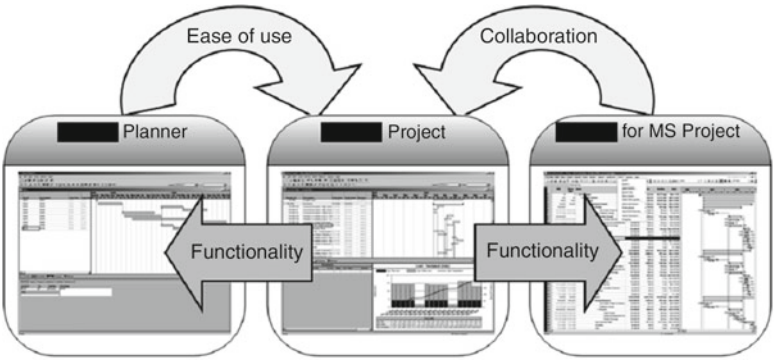

Fig. 7.5 The company's presentation of the relationship between the three products: Planner, Project and MPX (company's extension to MS Project). Planner is a standalone application and MPX a web applicaton

The respondent explains how a major request for change led from "user option" or "patch" and eventually "became part of the product." When a patch, provided for a few customers, is included in a later release cycle of a product, it will become "part of the product" and available to all customers.

The next excerpt is a follow-up question that illustrates the more common situation with regard to adaptation, namely a specific solution for one customer.

Interviewer: So, the rationale for a given upgrade lies with a specific customer, which means that a customer can be a part of setting the standards for what other customers receive?

Respondent: Mm, but if what one customer suggests is far off, then we just make a local adaptation for that specific customer.

The process of incorporating a customer's solution and making it part of the product or spawning new products does not occur very often, because the changes may infringe on property rights. Such changes are mainly initiated by large customers or supported by contracts, such as the engineering company, the oil company and the competitor depicted in Fig. 7.4. In most cases, improvement requests and end-user development activities are responded to by an adaptation, which means a custom-made solution for a single customer created by the company by using patches or user options within the current version of the product.

The idea behind the company's first product (Project) began in 1993. This idea has been preserved and refined in the latest products (Planner & MPX). The first product is, in effect, a template for all subsequent products and the standard against which new products are measured. One consultant was asked to describe what was meant by this and he explained it with a specific example: *"About 4 years ago, we extracted the interactive Gantt drawing part of Project and generated a new low-end bar chart drawing tool named Planner."* Figure 7.5 shows the relationship between the three products. Planner and MPX extend the core design idea of Project

(a technique for plotting diagrams) in two directions (ease of use with Planner and increased collaboration with MPX). This ensures a common strategy of development and the continued recognition of the company's products by customers.

7.7.3 Analysis

The Gantt drawing feature of Project is a predecessor artifact of both Planner and MPX, and is visible to those who know these products. This feature has been sustained for three successive products over a period of almost 20 years. It is a reusable design concept, i.e. a generalization of variations of a feature that originated in a specific implementation. The data from the case study shows that customers have proposed some of the features the company has incorporated in their products. The path from adaptation to generalization is complex and subject to multiple factors: proposals for new ideas, acceptance with or without payment, the different stages through which an innovation must pass (user option, patch, build, version, etc.). Only those proposals judged to be "good enough" would pass through the loophole and be taken further; the others will be rejected or will require payment by customers (Andersen and Mørch 2009). There is both intimacy and risk associated with joining forces in generalization: the customers want the best possible tools for their project-planning work, and the company wants to increase revenue and do the things they are good at (writing project planning software). These two processes are tension-laden (e.g. infringe on property rights) and predecessor artifacts emerge in this tension (Mørch et al. 2009), simultaneously addressing specific and general needs. PAs are represented in the specialized products by a reusable design concept important to the company (diagram plotting) that serves as a *reminder* of a feature that was first introduced in an older product (interactive Gantt drawing module of Project).

7.8 Summary and Discussion

Three types of predecessor artifacts have been described in this chapter based on examining data from previous projects:

1. *Reusable design material*: A predecessor artifact is represented by a physical trace of previous functionality (reuse of objects and concrete classes in BasicDraw to create KitchenDesign)
2. *Alternative functionality*: A predecessor artifact becomes a backup system (multiple alternatives to using a computerized information display)
3. *Reusable design concept*: A predecessor artifact is represented by an abstract reminder (an important design concept in Project was preserved and refined across its product line)

7.8.1 *Predecessor Artifact as Reusable Design Material*

George Herbert Mead, the philosopher and social psychologist, was among the first to recognize that there is a connection between emergence and creativity. When describing emergence he emphasized interaction among the parts and the whole and a reconstructing that is not given in advance, but necessary for seeing more than the sum of parts (Mead and Morris 1934). In the design computing community some researchers have developed a theoretical relationship between emergence and creativity, and define emergence "when a new form or concept appears that was not directly implied by the context from which it arose" (Seevinck and Edmonds 2008, p. 541). For example, two overlapping squares may reveal a triangle, which cannot be found by looking at the squares in isolation. Interactive systems can provide opportunities for creative interaction by exploiting emergence, enabling users to see something new in older things (Seevinck and Edmonds 2008), or to see something "old" in current things as profiled in this chapter. For example, the old wooden beams in the new house displayed in Fig. 7.1a refer to an older farmhouse on the same site, a physical trace of a predecessor artifact. This information provides viewers with an opportunity to learn something new, the evolutionary history of a certain building. At a more general level it points toward a conceptual framework for learning on demand based on phenomenology (the "extended present"). This framework is outside the scope of the present study, and suggests an area for further work.

In the first study, the rectangle was used as an emergent shape in the design units of KitchenDesign (see Fig. 7.2). This type of emergence is called *perceptual emergence* (Seevinck and Edmonds 2008). It is a relationship of two objects (in this case a rectangle and a kitchen cabinet symbol) as perceived by the users. The different abstraction levels of a computer application enable *perceptual* (user interface), *conceptual* (program code), and *physical* (run time organization) emergence. It is arguable program code that provides the most flexible level of reuse for computer applications (i.e. using the class/subclass relation in object-oriented programming), but software components can also provide flexibility in terms of cloning and integration with other components, without accessing source code (Mørch et al. 2004a, b).

A reusable design material can be compared with the design of new buildings by appropriating parts of older buildings and other reusable material in the built environment. For example the Container City in the London Docklands consists of buildings that uses recycled shipping containers for accommodation, resulting in a green and affordable solution to an inner city housing problem (Brand 1994). The original intention of a design may not be preserved when a new design is built from reusable design materials as opposed to when it is based on reusable design concepts. The reuse effect of appropriated software material is manifested in the run time system of an application (objects in memory) and in the user interface (objects on the screen), as well as in the minds of its designers. In program code it is manifested in subclasses (new code) that inherits from super classes (existing code).

7.8.2 Predecessor Artifact as Alternative Functionality

In his early but influential study of computer-supported work in organizations, Gasser (1986) identified a type of work he called *adaptation work*, which was composed of three types of coping with difficult-to-use computers in organizations: fitting, augmentation, and working around. *Fitting* is the strategy of modifying a computer system or changing the structure of work to accommodate a mismatch between workers and technology, i.e. a form of adaptation. *Augmentation* refers to undertaking additional work to make up for an inconsistency in primary work. *Working around* refers to using a computer system in ways it was not intended, or avoiding its use and relying instead on alternative, suboptimal means. One example is *backup systems* (Gasser 1986), which are older technologies one relies on when the main work support fails or becomes temporarily unavailable. An example of a backup system is Post-It notes around a computer display in order to remember difficult operating system commands (see Fig. 7.3a).

A backup system is a type of predecessor artifact that provides alternative functionality, but it is different from the two other types of PAs profiled in this chapter. It is a proper PA, functional artifacts that provide suboptimal but "good enough" alternatives to a new innovation. An example is banking in Norway. Although it is possible to visit a physical bank and make transactions by interacting with a teller, the teller will charge a fee for the service (suboptimal). Internet banking on the other hand is cheaper but requires digital competency. Another example is coin-operated telephones, serving as an alternative to a mobile phone. Most of the time people do not use these telephones, but they can be surprisingly useful when one's mobile phone does not work.

The second case revealed a more subtle form of relation between two artifacts, multiple technologies for supporting information seeking. This is what Suchman referred to as *artful integration*, a hybrid of technology and work practices where technology is comprised of multiple layers of heterogeneous devices, each associated with a specific generation of work support (Suchman 1994). This relationship between two artifacts (new innovation and PA) is also called *perceived resemblance* (Mørch 2003). This refers to a subjective interpretation of the likeness of two artifacts, which may cause dispute regarding the reuse of material or ideas from one to the other (e.g. paper catalogs vs. computerized information displays). The weaker form of relationship has the advantage that the older technologies may coexist and provide alternative work support when the new technology temporarily fails. Despite this, most backup methods are best thought of as the *endangered species* of the evolving artifacts. For example, phone books are gradually disappearing in favour of online services of personal information. A company that wishes to spearhead a modern technology profile, like the one studied in the second case, will often strive towards removal of older technologies even though the older technologies can provide useful alternatives for some employees. It seems that older technologies that cater to multiple ways of use may enjoy a longer life span (e.g. Post-It notes).

7.8.3 *Predecessor Artifact as Reusable Design Concept*

Ehn applied Wittgenstein's concepts of language game and family resemblance as metaphors to explain the development and use of a newspaper publishing system at the time when typesetting became computerized in Sweden (Ehn 1988). When the system showed family resemblance to something the typesetters already knew well, it stimulated active user participation in the design process and avoided common pitfalls, such as failing to meet user requirements (Ehn 1988). The graphical user interfaces could remind them of previous experiences of type setting. The reusable design in that situation is abstract representations of older typesetting tools and practices associated with new computer-based tools. Due to their abstract nature, these symbolic reminders of PAs are more difficult to capture than the reusable design material described above, but in return they reveal a more durable (space conserving) structure and they apply in more than one situation.

The third study identified predecessor artifacts by analyzing critical events surrounding new releases of a product line spawned by a commercial software house. Familiarity with the original product (Project) made it easier to use the later products (Planner and MPX), and to migrate from one product to the next. This was possible due to the family resemblance evident in the three systems, according to a special technique for displaying diagrams.

Reusable design concepts can be compared with architectural design patterns and archaeological skeuomorphs. Alexander and colleagues have described 250 *design patterns* of buildings and urban spaces (Alexander et al. 1977). Patterns serve as the basic building blocks of problem solving, providing answers to recurring design problems (e.g. How many stories should a building have?). Alexander's pattern language and cataloguing efforts have inspired software engineers, and numerous software design patterns and pattern languages for software systems have been proposed.

A *skeuomorph* is a technical term archaeologists use to denote a non-functional feature of an artifact that was inherited from some precursor artifact (Basalla 1988). The role of skeuomorphs in the evolution of artifacts has been studied in archaeology (Basalla 1988) and architecture of houses (Brand 1994). The notion of skeuomorphs has not yet been widely adopted in software design. As a visible representation of previous functionality at an abstract level, computational skeumorphs can provide an alternative to design patterns for integrating general level design information with software artifacts. This identifies an area for further work.

Seeding, Evolutionary Growth, Reseeding (SER) is a process model for the evolutionary development of software applications by integrating end-user and professional development (Fischer et al. 1994). It postulates that systems that evolve over a sustained time span must continually alternate between periods of unplanned evolutions by end-users (*evolutionary growth*), and periods of deliberate restructuring and enhancement (*reseeding*), involving users in collaboration with skilled software developers. This corresponds roughly to the two interdependent processes of adaptation (evolutionary growth) and generalization (reseeding). However, there are

no equivalents of reusable design concepts or predecessor artifacts to account for the transition from one release of a software application to the next in the framework of Fischer et al. (1994).

7.9 Implications for Reflective Conversation with Design Materials

Schön was inspired by the language game metaphor of Wittgenstein when he called reflective conversation (drawing and talking) a language game for doing architecture (Schön 1983). However, Schön did not exploit the full power of the metaphor, since he did not adopt the notion of family resemblance. Wittgenstein (1967) explained the connection of elements in a language game according to the similarity of traits among family members, "sometimes overall similarities, and sometimes similarities of detail." Two forms of resemblance are suggested in order to capture this distinction: *self-resemblance* and *family resemblance*. Sometimes there are *similarities of detail* (self-resemblance); at other times there are *overall similarities* (family resemblances). This allows for a new design or current artifact to be connected with previous designs (PAs) in two different ways, i.e. reuse of material (like in case 1, self-resemblance, well-defined connection) and reuse of concepts (like in case 3, family resemblance, generally accepted but weaker connection). In addition, a third type of connection between a current artifact and PAs was explored in case 2, *perceived resemblance*, which is a subjective likeliness of two things, serving an important function by allowing breakdowns to be resolved by resorting to alternatives. This is a useful distinction for analyzing adaptations of information systems in multiple stages (from design to use), extending the vocabulary of Schön and Ehn. For example it can help to identify a poor organizational implementation of a computer application, or a software design that fails to build on best practice, or an application that fails because it is rejected in favor of simpler alternatives.

Predecessor artifacts can also help the designer to cope with complexity in information-rich environments by addressing the following questions: How does the designer know that information is needed when designing, and when is a piece of information (e.g. design knowledge) relevant to the situation at hand? It is not possible to keep all potentially relevant information in short-term memory; this will lead to information overload. The distinction between known and potentially relevant information was explored in the Janus design environment (Fischer et al. 1989) and operationalized as computer-based back talk (critique) to stimulate reflection-in-action (McCall et al. 1990).

The past work the author has been involved in with regards to Janus has been extended and generalized through the notion of reusable design concepts in this chapter. A reusable design concept is an abstraction (general-level information) originating as a feature in a specific predecessor artifact. An example from the kitchen design domain is the *work triangle*, a concept that refers to the relative position of sink, stove, and refrigerator in a household kitchen. In Janus this concept together with a set of related design principles were implemented in software as critique

(condition-action rules) for detecting suboptimal design situations. An application of this was to inform students about the design principles to be resolved in their own designs. In the work presented here the previous work has been generalized to mean "expansive design," identifying in current artifacts emergent properties (traces and reminders) that represent predecessor artifacts.

7.10 Conclusions and Directions for Further Work

This chapter has presented an evolving-artifacts approach to integrate creativity and rationale. The common denominator has been predecessor artifacts (PA) and resemblance relationships to connect PAs with newer artifacts. Schön's theory of reflective conversation with design material provided the starting point for this inquiry, identifying a shortcut in his account of evolutionary design (bypassing generalization and predecessor artifacts). The aim of this study has been to extend the vocabulary of reflective conversation and make it applicable to analyzing software applications that evolve. Three types of PAs were identified and differentiated based on system building efforts and empirical findings from three case studies (*reusable design material*, *alternative functionality,* and *reusable design concepts*), and three types of resemblance relations based on previous work and literature studies to bring the inheritance relation between class and subclass in object-oriented programming to a user oriented level of abstraction (*self*, *perceived, and family*), hence proposing a semi-formal approach to software reuse and design. Predecessor artifacts relate to creativity and rationale in the following manner. On one hand a predecessor artifact is a rejected alternative (*product-oriented rationale*) that represents the design history of a current artifact (*process-oriented rationale*). On the other, it is a concrete (physical) trace or abstract (symbolic) reminder in a new artifact (*an emergent property*), and as such can provide opportunities for creative interaction with an artifact, such as enabling users to see a previous version of a tool (a PA) in a new tool, and by repeating the process, the myriads of ways of connecting reusable designs.

This work has implications for discourse analysis methods in design (e.g. interaction analysis), suggesting a dimension of historical artifact analysis to supplement spoken dialogue and body language (deictic references and drawing). It can also provide ideas for design history education in schools of design (e.g. the role of traces and reminders in everyday artifacts), and design-based models of human learning and development in schools of education. In terms of information and computer sciences, this research suggests that some abstract features of today's software tools can be traced back to functional features in some predecessor artifacts. If this relationship can be made durable, it may help to create a tighter connection between program code and design patterns in software projects, and make "software archeology" a new research topic.

This work has not addressed: (1) A fourth type of PA, *disposable artifacts*, older tools of no practical or minimal symbolic value, often removed to reclaim occupied space, and (2) *successor tools*: contrasted with PAs as future tools anticipated and designed by improving the present.

Acknowledgements The author thanks the former InterMedia staff and students who contributed to the research: Camilla Brynhildsen, Bård Ketil Engen, Mari Ann Skaanes, and Ida Tødenes who studied the petrol station workers; and Renate Andersen, Shazia Mushtaq, and Kathrine Nygård who studied the interaction of professional and amateur software developers. Further, the author is grateful for Jack Carroll's invitation to submit this chapter to the Springer volume on creativity and rationale. The author received financial support from the Research Council of Norway (Learning at work project, 2001–2004), and the European Commission's program under Framework 6 (Knowledge practices, KP-Lab, project, 2006–2011).

References

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language: Towns, buildings, construction*. New York: Oxford University Press.

Andersen, R., & Mørch, A. I. (2009). Mutual development: A case study in customer-initiated software product development. In V. Pipek, M. B. Rosson, B. de Ruyter, & V. Wulf (Eds.), *Proceedings 2nd Int'l Symposium on End User Development (IS-EUD 2009), LNCS* (Vol. 5435, pp. 31–49). Berlin: Springer.

Ball, L. J., Lambell, N. J., Ormerod, T. C., Slavin, S., & Mariani, J. A. (2001). Representing design rationale to support innovative design reuse: A minimalist approach. *Automation in Construction, 10*, 663–674.

Basalla, G. (1988). *The evolution of technology*. Cambridge: Cambridge University Press.

Bergson, H. (1993). *Creative evolution*. Lanham: University Press of America (Original published by Henry Holt and Company, New York, 1911).

Brand, S. (1994). *How buildings learn: What happens after they're built*. London: Phoenix Illustrated.

Carroll, J. M. (1994). Making use a design representation. *Communication of the ACM, 37*(12), 28–35.

Carroll, J. M. (2011). Design rationale and appropriation: Providing resources to facilitate creative use. In *Proceedings CHI 2011, Workshop on appropriation and creative use*. New York: ACM.

Carroll, J. M., & Rosson, M. B. (1996). Deliberated evolution: Stalking the View Matcher in design space. In T. P. Moran & J. M. Carroll (Eds.), *Design rationale: Concepts, techniques, and use* (pp. 107–145). Mahwah: Lawrence Erlbaum Associates.

Conklin, E. J., & Burgess-Yakemovic, K. C. (1996). A process-oriented approach to design rationale. In T. P. Moran & J. M. Carroll (Eds.), *Design rationale: Concepts, techniques, and use* (pp. 393–427). Mahwah: Lawrence Erlbaum Associates.

Ehn, P. (1988). *Work-oriented design of computer artifacts*. Stockholm: Arbetslivscentrum.

Feyerabend, P. (1999). In B. Terpstra (Ed.), *Conquest of abundance: A tale of abstraction versus the richness of being*. Chicago: University of Chicago Press.

Fischer, G. (1994). Turning breakdowns into opportunities for creativity. *Knowledge-Based Systems, 7*(4), 221–232.

Fischer, G., McCall, R., & Morch, A. (1989). Janus: Integrating hypertext with a knowledge-based design environment. In *Proceedings Hypertext'89* (pp. 105–117). New York: ACM.

Fischer, G., McCall, R., Ostwald, J., Reeves, B., & Shipman, F. (1994). Seeding, evolutionary growth and reseeding: Supporting the incremental development of design environments. In B. Adelson, S. Dumais, & J. Olson (Eds.), *Proceedings of CHI'94* (pp. 292–298). New York: ACM.

Gasser, L. (1986). The integration of computing and routine work. *ACM Transactions on Information Systems, 4*, 205–225.

MacLean, A., Young, R., Bellotti, V., & Moran, T. (1991). Questions, options, and criteria: Elements of design space analysis. *Human Computer Interaction, 6*(3–4), 201–251.

McCall, R. (1991). PHI: A conceptual foundation for design hypermedia. *Design Studies, 12*, 30–41.

McCall, R. (2010). Critical conversations: Feedback as a stimulus to creativity in software design. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments, 6*(1), 11–37.

McCall, R., Fischer, G., & Morch, A. (1990). Supporting reflection-in-action in the Janus design environment. In *The electronic design studio* (pp. 247–259). Cambridge, MA: MIT Press.

Mead, G. H., & Morris, C. W. (Eds.). (1934). *Mind, self, and society*. Chicago: University of Chicago Press.

Moran, T. P., & Carroll, J. M. (Eds.). (1996). *Design rationale: Concepts, techniques, and use*. Mahwah: Lawrence Erlbaum Associates.

Mørch, A. (1996). Evolving a generic application into a domain-oriented design environment. *Scandinavian Journal of Information Systems, 8*(2), 63–90.

Mørch, A. (1997). Three levels of end-user tailoring: Customization, integration, and extension. In M. Kyng & L. Mathiassen (Eds.), *Computers and design in context* (pp. 51–76). Cambridge, MA: MIT Press.

Mørch, A. I. (2003). Evolutionary growth and control in user tailorable systems. In N. V. Patel (Ed.), *Adaptive evolutionary information systems* (pp. 30–58). Hershey: IGI Publishing.

Mørch, A. I., & Skaanes, M. A. (2010). Design and use of an integrated work and learning system: Information seeking as critical function. In S. Ludvigsen, A. Lund, I. Rasmussen, & R. Säljö (Eds.), *Learning across sites: New tools, infrastructures and practices* (pp. 138–155). London: Routledge.

Mørch, A. I., Engen, B. K., & Åsand, H.-R. (2004a). The workplace as a learning laboratory: The winding road to e-learning in a Norwegian service company. In A. Clement et al. (Eds.), *Proceedings of PDC'2004* (pp. 141–151). New York: ACM Press.

Mørch, A. I., Stevens, G., Won, M., Klann, M., Dittrich, Y., & Wulf, V. (2004b). Component-based technologies for end-user development. *Communications or the ACM, 47*(9), 59–62.

Mørch, A. I., Nygård, K. A., & Ludvigsen, S. R. (2009). Adaptation and generalisation in software product development. In H. Daniels et al. (Eds.), *Activity theory in practice: Promoting learning across boundaries* (pp. 184–205). London: Routledge.

Norman, D. A. (2008). Workarounds and hacks: The leading edge of innovation. *Interactions, 15*(4), 47–48.

Petroski, H. (1992). *The evolution of useful things*. New York: Vintage Books.

Rittel, H. (1972). On the planning crisis: Systems analysis of the first and second generations. *Bedriftsøkonomen, 8*, 390–396.

Schön, D. A. (1983). *The reflective practitioner: How professionals think in action*. New York: Basic Books.

Schön, D. (1992). Designing as a reflective conversation with the materials of a design situation. *Knowledge-Based Systems, 5*(1), 3–13.

Seevinck, J., & Edmonds, E. (2008). Emergence and the art system 'plus minus now'. *Design Studies, 29*(6), 541–555.

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). Cambridge, MA: MIT Press.

Suchman, L. (1994). Working relations of technology production and use. *Computer Supported Cooperative Work, 2*(1–2), 21–39.

Tomlinson, B. (2008). A call for pro-environmental conspicuous consumption in the online world. *Interactions, 15*(6), 42–45.

Wittgenstein, L. (1967). *Philosophical investigations* (3rd ed.). Oxford: Basic Blackwell.