

Debugging an Artifact, Instrumenting a Bug: Dialectics of Instrumentation and Design in Technology-Rich Learning Environments

Tobin White

Published online: 19 July 2007
© Springer Science+Business Media B.V. 2007

Abstract This article explores ways of conceptualizing the design of innovative learning tools as emergent from dialectics between designers and learner-users of those tools. More specifically, I focus on the reciprocities between a designer's objectives for student learning and a user's situated activity in a learning environment, as these interact and co-develop in cycles of design-based research. Recent investigations of technology-supported mathematics learning conducted from an 'instrumental' perspective provide a powerful framework for analyzing the process through which classroom artifacts become conceptual tools, simultaneously characterizing the ways students come to both implement and understand a device in the context of a task. Similarly, design-based approaches to investigating instructional activity offer epistemological grounds for treating the process of designing artifacts to support learning as unfolding in concert with rather than concluding prior to situated student use. Drawing on each of these perspectives, I describe the design and initial implementation of a set of software artifacts intended to support students' collaborative problem solving through locally networked handheld computers. Through detailed analyses of three classroom episodes, I report on the ways one student group's innovative and unexpected use of these tools served as an opportunity to both examine student learning in the context of that novelty and to refine the software design. This account provides an empirical example through which to consider the potential for instrumental genesis to inform design, and for design research epistemology to broaden the scope of instrumental theory.

Keywords Artifacts · Design research · Functions · Instrumental genesis · Multiple representations

T. White (✉)
School of Education, University of California, Davis, One Shields Avenue, Davis,
CA 95616, USA
e-mail: twhite@ucdavis.edu

1 Introduction

Investigations of technology-in-use in a variety of contexts have stressed that users in a particular setting negotiate meanings for a tool that can be quite different from those attributed by designers prior to the artifact's arrival in that setting (Suchman 1986; Suchman et al. 1999). For example, the study of organizations has helped to show the dual nature of technologies as both constituted by and constitutive of social practices (Orlikowski 1992). Researchers examining the new arrival of desktop computers in classrooms have similarly suggested that teachers and students organize the role of the tool in ways that interact with but are not determined by any of the tool's inherent features (Mehan 1989). Together, these various portrayals illustrate the practices associated with tools as deeply situated, arising in distinctive and sometimes surprising ways from the settings in which they occur.

This situated character of technological meaning has important implications for the design of new classroom artifacts. If the application of a tool is inevitably underdetermined by its design, specifying the ways that tool might support learning will require careful consideration of its use by students in instructional activities and contexts. Though the complexities associated with anticipating tool use pose challenges for design, they may also constitute opportunities. Learners' capacities to craft creative and unexpected mathematical meaning from and with tools may provide a powerful resource for informing revision or identifying alternative implementations. Indeed, unanticipated tool uses and task environments in which they proliferate may represent design objectives in their own right.

This article explores ways of conceptualizing the design of innovative learning tools as emergent from dialectics between designers and learner-users of those tools. More specifically, I focus on the reciprocities between a designer's objectives for student learning and a user's situated activity in a learning environment, as these interact and co-develop in cycles of design-based research. Recent investigations of technology-supported mathematics learning conducted from an 'instrumental' perspective provide a powerful framework for analyzing the process through which classroom artifacts become conceptual tools, simultaneously characterizing the ways students come to both implement and understand a device in the context of a task. Similarly, design-based approaches to investigating instructional activity offer epistemological grounds for treating the process of designing artifacts to support learning as unfolding in concert with rather than concluding prior to situated student use. These perspectives each attend carefully to the situated and emergent meaning of instructional artifacts, but from different analytic standpoints, one focused on the ways a learner makes sense of an artifact and the other on ways of designing artifacts as resources for making sense of learners. The objective of this article is to explore potential synergies between instrumental genesis theory and design research methodology with an eye toward enriching both approaches. After outlining each of these perspectives in detail, I describe a set of software artifacts designed as tools to support students' collaborative problem solving with networked handheld computers. I then present a sequence of three classroom episodes through which a student group's innovative and unexpected use of these tools served as an opportunity to both examine student learning in the context of that novelty and to refine the software design.

2 Theoretical and Empirical Perspectives

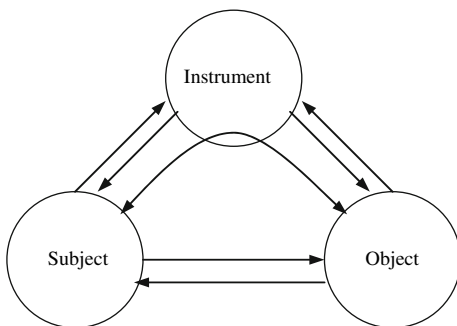
2.1 Instrumental Genesis

Verillon and Rabardel's (1995) account of instrumental genesis, the process through which artifacts become tools to be utilized in the accomplishment of a task, provides a powerful analytic framework for examining student learning in technology-rich environments. From this perspective, an instrument represents the union of an objective artifact, such as a physical device or software component, with a particular user's conceptual scheme for implementing that artifact in a specific activity situation. Instruments thus emerge through a dialectical interplay between the technical demands of mastering a device and the conceptual work of making that device meaningful in the context of a task (Artigue 2002). This linking of the technical and the conceptual offers considerable utility for analyzing mathematics learning with classroom technologies, as evidenced by a number of recent investigations making use of the instrumental genesis perspective to examine tools such as graphing calculators (Guin and Trouche 1999; Lagrange 1999), computer algebra systems (Artigue 2002; Ruthven 2002), spreadsheets (Haspekian 2005), dynamic geometry environments (Mariotti 2002) and dynamic simulations (Hegedus 2005).

The dialectic between conceptual and technical aspects of tool use in instrumental genesis unfolds through the intertwined processes of instrumentalization, oriented toward the artifact, and instrumentation, oriented toward the user (Trouche 2004). Through instrumentalization, an artifact becomes a means of achieving an objective, solving a problem, completing a task—it becomes meaningful to an activity situation, and thus has been transformed into an instrument. This transformation of the artifact pairs with the simultaneous transformation of the user, as through instrumentation the user develops the schemes and techniques through which the artifact can be implemented in purposive action. As Trouche remarks, "*instrumentation* is precisely the process by which the artifact *prints its mark* on the subject... One might say, for example, that the scalpel *instruments* a surgeon" (p. 290, emphasis in original). Instrumentation involves forming a *utilization scheme* that provides a predictable and repeatable means of integrating artifact and action (Verillon and Rabardel 1995). Trouche emphasizes that utilization schemes comprise not only the rules and heuristics for applying an artifact to a task, but also the understanding of the task through which that application becomes meaningful to the user. That understanding takes the form of "theorems-in-action" which specify the knowledge underlying a particular scheme. In classroom mathematical activity, theorems-in-action take shape as the domain-specific propositions on which learners rely as they interpret the capabilities and constraints of a tool in relation to the features of a problem-solving task. Herein lies the particular utility of the instrumental genesis perspective for classroom research: hypothesizing about the theorems-in-action guiding a learner's engagement with tasks and tools provides a mechanism for linking that learner's instrumented activity with learning goals and curricular content.

Instrumental genesis thus both makes artifacts meaningful in the context of activity, and provides a means by which users make meaning of that activity. This perspective is compatible with other recent accounts of classroom devices (e.g. Meira 1998), and has its origins in the cultural-historical theory of Vygotsky (1978), in which human activity is distinguished by the mediating role of symbolic artifacts in regulating "higher psychological processes." Mediating artifacts both constitute and are constituted through activity; an artifact is imbued with meaning—shaped as an instrument—through its implementation

Fig. 1 The instrumented activity system model (Verillon and Rabardel 1995)



in a specific task, toward a particular end. Correspondingly, the study of an instrument is the study not of an object, but of a process, the genesis of its significance to a particular user and for a particular purpose. Figure 1, from Verillon and Rabardel (1995), illustrates the triadic relationships between subject, object and mediating artifact that comprise an instrumented activity system¹.

This emphasis on the process through which the meaning of an artifact is constituted through goal-oriented activity has important methodological consequences. From this perspective, examining processes of instrumental genesis in classroom mathematical activity amounts to a microgenetic analysis of the mathematical meaning of an instrument in relation to a problem-solving task. Guin and Trouche (1999) have argued that the analysis of learners' instrumented activity can contribute to the design of instruction by highlighting the mathematical knowledge involved in its use. In a similar spirit, analyzing the processes through which instruments emerge may shed additional light on the forms of mathematical learning they support.

Importantly, those processes take time; prior work on the instrumental genesis of classroom artifacts rightly emphasizes the length and complexity of the process through which sophisticated devices become meaningful mathematical tools for students (Artigue 2002). Moreover, the path from artifact to instrument is unlikely to unfold as a simple linear trajectory; Verillon and Rabardel (1995) note that the microgenesis associated with learning to apply a tool to a task is not likely to be "a simple process of the subject's assimilation-accommodation of an artifact," but rather "a double elaboration that is both progressive and interdependent: that of the properties of the technical system, and that of the properties of the reality to be transformed" (p. 25). In searching for effective and efficient means of implementing an artifact, learners can be expected to progressively reinterpret both the emerging instrument and the activity situation, gradually and successively revising their utilization schemes as they work to more effectively orient their instrumented actions toward the accomplishment of the task. As such, the emergent negotiations through which learners instrumentalize and are instrumented by devices may reveal as much about their developing conceptions of the objects on which they act—in the form of unfolding theorems-in-action underlying those successive schemes—as of the instruments that mediate those actions.

¹ Similar triangular diagrams play a central role in the account of mediated activity provided by cultural-historical activity theory (Cole and Engeström 1993; Cole 1996). These respective models derive from the same Vygotskian origins, but differ in terms of analytic focus; while the activity-theoretic perspective emphasizes the constitutive role of artifacts in human activity, the Verillon–Rabardel framework elaborates the ways mediating artifacts are constituted through activity systems.

2.2 Tool Design and Design Research

The theory of instrumental genesis emphasizes the nature of instruments as fluid and emergent phenomena, the meanings of which are negotiated and renegotiated by particular users in the context of particular activities. By contrast, the artifacts themselves remain static, even as they may be instrumentalized in very different ways. While an artifact might become a very different instrument for one user or task than it had been for another, those changes do not entail any necessary changes to the physical characteristics or structural properties of the artifact itself. This characterization of artifacts should certainly prove adequate to the investigation of classroom tools such as commercially available calculators and computer software packages, which can be expected to remain stable over the course of their classroom use and across different classroom settings.

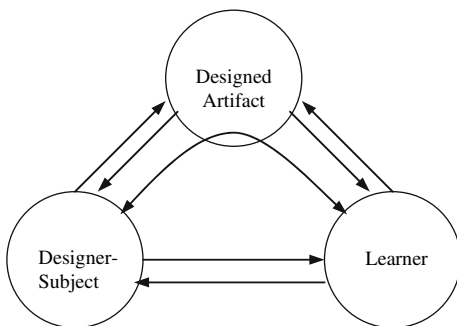
But those classroom artifacts invariably have histories of their own; they were shaped with purpose, and for purposes, perhaps including ends quite different from those to which they might be put by students and teachers. Yerushalmy (1999) notes that “the design of a tool reflects the designer’s intentions, thoughts, and compromises. Design represents a point of view and reflects in some way the purpose for which the tool was created” (p. 171). Thus while the meaning of an instrument emerges from the aim toward which a given subject applies an artifact, that artifact also embodies an objective envisioned by the designer. For example, Yerushalmy distinguishes between tools common to the mathematics classroom that were intended to provide efficient solutions to problems, and those designed to support the exploration of curricular concepts. These different intentions both enable and constrain learner actions in quite different ways. Consequently, instruments can be seen as always jointly emerging from the aims of two subjects, a designer and a user.

Moreover, the arrival of a device in the classroom need not mark the conclusion of a design process. Indeed, many contemporary research efforts focus on classroom technologies that are still in development, and even emphasize successive designs as a methodological principle. The still-emerging paradigm of design-based research takes the iteration of theory-driven development, classroom deployment, and data-driven redesign cycles to be one of its hallmarks (Design-Based Research Collective 2003). Design research in education treats technological, pedagogical and curricular innovations as opportunities to investigate the new forms of learning they facilitate and make salient (Cobb et al. 2003). The very novelty of these innovations makes the new phenomena to which they give rise at once both possible and unpredictable; early efforts to implement new designs will inevitably yield some unexpected results. Successive iterations of implementation and redesign allow for the new insights gleaned from such results to be incorporated into the progressive refinement of the design “until all the bugs are worked out” (Collins et al. 2004, p. 18).

But in design research, the process is as important as the products, and sometimes the bugs can become the phenomena of interest. diSessa and Cobb (2004) draw on two case studies to illustrate the process of “ontological innovation” through which design researchers developed explanatory constructs in order to accommodate the emergence of unexpected phenomena during classroom investigations. In each case, researchers drew on retrospective analyses of video and other data to make sense of a surprising occurrence, and ultimately reorganized their immediate and subsequent designs and investigations around the new constructs and categories spawned by those analyses.

The unexpected occurrence I will analyze in this article was more modest in its contributions to ontological innovation than those presented by diSessa and Cobb. The immediate phenomenon revealed by the event involved a particular software design element, namely a bug in the program, rather than an aspect of student learning or classroom

Fig. 2 Design research as instrumented activity



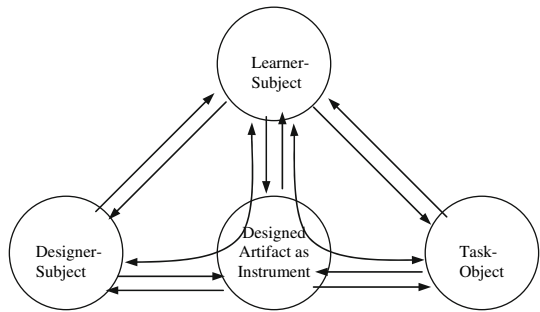
activity. As such, it presented a complex set of technical, pedagogical, and logistical challenges and questions relating to whether, how and when the bug should be fixed in the midst of a rapid instructional sequence. Within a few days, and through the frantic efforts of several research team members, the software was revised and the devices updated, and instruction continued virtually uninterrupted. However, subsequent and more detailed retrospective analysis of this event, which I present below, revealed additional and more intriguing layers to the phenomenon. While the discovery of this software bug was both a surprise and a crisis to be resolved, the more significant revelation involved the ways that a group of students, unaware that it was anything other than an intended feature of the software, began treating the bug as an affordance of a problem-solving tool. The resulting story provides a way to examine the relationships between the design of artifacts and the genesis of instruments, particularly as these processes might productively intertwine in support of student mathematical learning. The next section proposes a framework for conceptualizing the reciprocities between the instrumentation and design of instructional artifacts.

2.3 Dialectics of Instrumentation and Design

In the previous section, I argued that the meaning of classroom mathematical instruments might be seen as jointly figured by the intentional activity of a designer² and a learner-user. This section elaborates that argument to further explore the ways a focus on design might broaden the scope of instrumental genesis theory, and the reciprocal contributions of that theory to the epistemology of design-based research. The instrumented activity system model of Fig. 1 highlights the ways a mediating artifact can transform subject, object, and the relations between them. In the case of instructional devices, the relevant subject is presumably a learner oriented toward a problem-solving objective or other classroom task. But such devices may also and simultaneously be understood as serving to mediate another subject-object dynamic, as a designer or educator instruments a designed artifact to accomplish instructional objectives oriented toward a learner (Fig. 2).

² My use of the term “designer” here is specific. I primarily have in mind design researchers, for whom designed artifacts serve the dual purposes of supporting and investigating student learning. I have used the broader term, however, in order to be inclusive of any developer of learning tools, or any educator who crafts instructional tasks around such tools—anyone for whom those tools might serve as resources for both bringing about and learning from instances of learner’s instrumented activity.

Fig. 3 Dialectics of instrumentation and design



As in the case of a learner's instrumented activity, the instructional implementation of a designed artifact might be understood both in terms of instrumentalization, as the designer clarifies the meaning of the tool in relation to desired learning objectives for the student, and in terms of instrumentation, as the designer's view of the learner evolves in relation to the latter's engagement with the artifact. In this way, the dual orientation of the artifact toward both subject and object characteristic of instrumental genesis is also a hallmark of design-based research, as the latter seeks to both catalyze learning experiences for the student, and inform the design researcher's understanding of that student's learning.

In design research efforts involving innovative tools, the same artifacts simultaneously mediate the researcher-designer's engagement with a learner and the learner's engagement with the object of an educational task. Figure 3 merges the triangles of Figs. 1 and 2 in order to depict the dual meaning of these designed instructional artifacts, constituted in relation to both the task-oriented activity of learner-users and the instructional and empirical objectives of design researchers. Fully apprehending the meaning of these instruments requires attending to these dual roles as they emerge in relation to one another. In particular, the effectiveness of designed artifacts for supporting student learning may ultimately depend on the extent to which a designer-educator's instructional objectives and a user-learner's task-orientation—the instructional and instrumental meanings of an artifact—can be brought into alignment.

3 The *Code Breaker* Learning Environment

3.1 Design Features: Linking Multiple Roles and Representations

This article draws on data collected during the first classroom implementation of a learning environment situated in a classroom network of wireless handheld computers. Intended to support mathematical learning through collaborative problem-solving activities, this designed environment attempted to capitalize on two features of the handheld devices: their capacities to simultaneously display multiple linked representations of a mathematical function, and to connect multiple students through a local wireless device network. A handheld client application, called *Code Breaker*, allowed students to edit parameters of a polynomial function, and to examine corresponding changes in a linked array of graphical, tabular and numerical displays, each of which provided different resources relevant to an open-ended problem-solving task. Through a desktop server application, the teacher assigned the devices of students who were seated together in groups of four to a corresponding server-defined group. Changes to the function on one student's handheld

automatically propagated to the devices of the other group members. Consequently, though a single device could display only one or two of those linked artifacts at a time, a group could collectively examine the full array of representations simultaneously.

A curricular unit accompanying this handheld network asked students to imagine themselves as cryptographers, and to collaborate with the other members of their small group on daily problem-solving activities involving the making and breaking of codes. To generate these codes, letters in the alphabet were assigned to their ordinal values 1–26, and then mapped through a polynomial function to produce a set of output values comprising a numerical cipher text alphabet. Decryption activities commenced when a group downloaded a string of numbers representing a message that had been encrypted by the teacher or by another student group. The problem-solving process involved using the *Code Breaker* software to match an editable ‘candidate function’ to the unknown ‘encoding function’ from which the encrypted message had been generated.

The *Code Breaker* interface allowed students to choose from among an array of representational views displaying elements of the candidate function and the encrypted message. These representations included a symbolic expression, cipher- and plaintext displays, a graph, function tables, and letter and word frequency charts. Together, these artifacts provided a set of resources students could collectively employ to analyze the mathematical properties of the candidate and encoding functions in order to decode a message. The characterization of representations in this environment as ‘artifacts’ requires some clarification. Because these representations were linked both within and across devices, and because solving decryption problems invariably required groups to coordinate multiple representations, the representations constituted both stand-alone artifacts that students applied to the analysis of codes, and also components of a distributed software artifact that the group collectively brought to bear on code-breaking tasks. The episodes below highlight individual representational artifacts as emerging mathematical instruments; importantly, however, these processes also intertwined in the genesis of a collaborative and multi-representational code-breaking tool. Five *Code Breaker* representational artifacts that figure centrally in the analyses presented in this article are described in detail below.

3.1.1 The Candidate Function

Both the candidate and encoding functions were always of the form $y = ax^b + c$, where c could be any integer, a any non-zero integer, and b could equal one, two, or three. One student in each group was assigned responsibility for editing the candidate parameters from their default settings ($a = 1$, $b = 1$, $c = 0$) by tapping on either the top or the bottom half of the number on their handheld screen, causing the value to increment or decrement by one unit. This student was referred to as the group’s ‘publisher,’ and the role was rotated among group members daily.

3.1.2 The Inverse Function Table

The instructional intent behind the inverse function table was to illustrate the bidirectional flow of functional relationships, so that if encoding a message creates a mapping between plain and cipher text, then decoding that message amounts to inverting that function by mapping the outputs back to their original inputs. The range of values in an encrypted message is displayed in the Y-column of the inverse function table, as shown in Fig. 4 for an encoding function of $y = 5x + 7$. Each of these cipher text values is then mapped through the inverse of the current candidate function ($y = 17x - 29$, in the example of Fig. 4), with the result

Fig. 4 The inverse function table

Letter	X	Y	Letter	X	Y
C	2.4	12		5.6	67
	2.7	17		5.9	72
	3	22		6.5	82
	3.6	32		6.8	87
	4.2	42		7.7	102
	4.5	47	H	8	107
	4.8	52		8.3	112

displayed in the corresponding cell of the X-column. When this process yields an integer from 1 to 26, the corresponding plaintext letter appears in the “Letter” column.

The version of *Code Breaker* implemented in this study featured an overlooked software bug with significant implications for the behavior of the inverse function table, and for the cases presented below. In order to provide additional code-breaking utility, the inverse mapping through the candidate function was designed to include a limited tolerance within which values that were nearly but not exactly integers from 1 to 26 were rounded to those integers. This rounding feature was intended to apply only to relatively large cipher text values generated by fairly complex quadratic and cubic encoding functions. In fact, a late revision of the rounding feature had the unexpected side effect of including *all* non-integer values with a greatest integral part between 1 and 26 within the rounding tolerance. In linear cases where the coefficient of the candidate function was greater than that of the encoding function, the table displayed multiple cipher text values mapped to the same letter. Figure 5 reveals what the ‘buggy’ table actually displayed, given the encoding and candidate functions from Fig. 4.

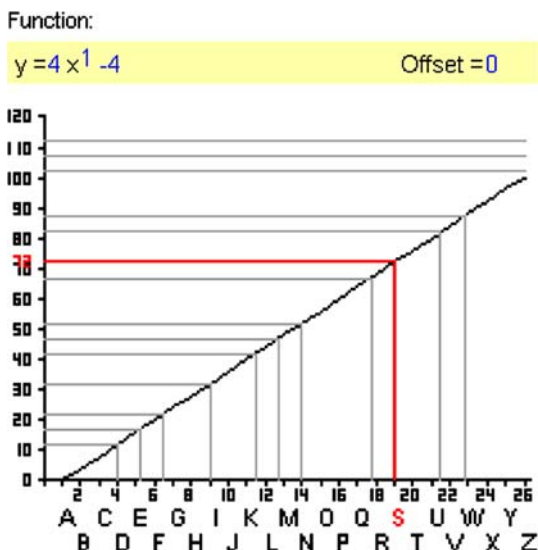
3.1.3 The Graph

The *Code Breaker* graph displays the candidate function curve in a window scaled in accordance with the encoding function. The x-axis of the graphing window is always fixed to the alphabetic domain, spanning from 0 to 26. The y-axis, on the other hand, adjusts automatically around the range of values included in the coded text message. Each of those cipher text values is also represented in the graph by a horizontal line stretching from the y-axis until it intersects the candidate curve. A corresponding vertical line, drawn from this intersection to the x-axis, reflects the mapping of the cipher text value through the inverse

Fig. 5 The inverse function table (with rounding bug)

Letter	X	Y	Letter	X	Y
B	2.1	12	E	5.1	67
B	2.1	17	E	5.1	72
C	3	22	F	6.1	82
C	3.1	32	F	6.1	87
D	4.1	42	G	7.1	102
D	4.1	47	H	8	107
D	4.1	52	H	8.1	112

Fig. 6 The graph



of the candidate function. When the output of this inverse mapping is an integer between 1 and 26, students can click on the associated letter and highlight the trace lines and the ordered pair they connect, as shown in Fig. 6.

3.1.4 The Function and Frequency Tables

The function table is one of the representations that is based entirely on the candidate function. Shown in Fig. 7, this table shows a static X-column with the numbers 1–26, and a dynamic Y-column that updates with each adjustment of the candidate function, showing the set of numbers to which that polynomial would map each of the X-values. As shown in Fig. 8, the frequency table is composed of three columns (as with the function and inverse function tables, these columns are divided in half, with the two halves placed side-by-side to preserve screen space). The Y-column displays all the numbers that appear in the coded text of the encrypted message currently being decoded. The “Count” column specifies the

Fig. 7 The function table

Function Table:

X	Y	X	Y
1	26	14	49,400
2	152	15	60,758
3	494	16	73,736
4	1,160	17	88,442
5	2,258	18	104,984
6	3,896	19	123,470
7	6,182	20	144,008
8	9,224	21	166,706
9	13,130	22	191,672
10	18,008	23	219,014
11	23,966	24	248,840
12	31,112	25	281,258
13	39,554	26	316,376

Fig. 8 The frequency table**Frequency Table:**

X	Y	Count	X	Y	Count
2	513	1		130,321	3
	152	1		19	4
	4,104	1		13,851	4
	25,289	1		110,808	4
	175,959	2		1,216	6
	6,517	2		152,000	6
	32,832	3		52,136	7
	231,173	3		2,375	7
	9,728	3		64,125	10

number of times that each of those coded text values appears in the encrypted message. Importantly, the table is sorted according to the count, from least to most frequent, rather than according to the relative numerical values of the coded text characters in the Y-column.

3.2 Description of the Study

The *Code Breaker* environment was implemented in a middle school mathematics classroom during a 5-week summer school session. Lessons and activities during the first week of the instructional unit introduced the function concept and the cryptography context, and students were oriented to the handheld computers and the *Code Breaker* software during the second week. Equipped with these resources, students spent the remaining three weeks collaborating in small groups to make and break codes. As they grew familiar with the different representational artifacts in the *Code Breaker* software, students in each group were encouraged to work together to develop their own strategies for using those resources to solve increasingly challenging decryption problems.

The analysis below will examine a series of excerpts from one small group's problem-solving activities with the *Code Breaker* handheld network. This group was videotaped each day during the final three weeks of the instructional unit. These video records, along with server logs, researchers' observation notes, and students' written records, were analyzed with regard to the problem-solving strategies enacted and the mathematical ideas expressed by the students in each decryption activity. The group was comprised of four students: CJ, Jason, Reggie, and Vince. These students were quite diverse in terms of prior mathematical achievement, and the nature and extent of their contributions to the collaborative activities varied widely. Nonetheless, for the purposes of this article I treat the group as the relevant unit of analysis; the group was the level at which tasks were undertaken, at which multiple representational artifacts were collectively applied to those tasks, and at which the negotiations of instrumental meaning presented here were made visible through student discourse. Variations among the experiences of students within groups in this environment are examined in detail in White (2006).

4 Analysis

I analyze three successive episodes in the unfolding process of instrumental genesis through which one student group came to use *Code Breaker* as a tool for solving decryption tasks. The handheld devices in the *Code Breaker* environment presented

students with a network of linked but distinct representational artifacts, each with different affordances and constraints relative to the decryption tasks. Consequently, the instrumental process involved negotiating meanings and uses for each of these artifacts in relation to the task. Each of the three episodes highlights one or more different representational artifacts, and explores the group's fashioning of those representations into decryption instruments. Episode one depicts the group's unexpected instrumentation of a software bug underlying the inverse function table. Episode two reveals the path the group followed, through that same bug, from their use of the inverse function table toward a preference for the graph. In the third episode, the group worked to coordinate a familiar representational artifact, the frequency table, with the new function table added during a redesign.

In keeping with the instrumental approach, I present each of these episodes in two parts. The first part of each case analysis emphasizes instrumentalization, detailing the group's efforts to direct the representational artifact toward the decryption task. The second part focuses on instrumentation, exploring the apparent relations between students' utilization of the artifact and their emerging understanding of the function concept. Of the three episodes, only the first endeavors to present a comprehensive analysis of instrumental genesis. The bug as instrument is of particular interest and importance to the argument of this article; moreover, in contrast to the other representational tools, use of the software bug was confined to a sufficiently small range of uses over a relatively short span of time, and thus more amenable to full consideration within the scope of the article. The other two episodes are not intended to fully represent the ways the group applied and understood the graph or the function and frequency tables relative to the constraints and affordances of those artifacts. Rather, they offer snapshots of significant moments in the genesis of these respective instruments in order to both provide contrasts to, and to complete the story of, the inverse function table bug. All three cases are linked through both the group's efforts to instrument the software to solve decryption tasks more effectively and through changes in the software design prompted by this group's work. After presenting these three episodes, I discuss their linkages from the alternate perspectives of instrumental genesis and design research.

4.1 Episode One: Instrumenting a Bug

This episode details the first recorded appearance of the rounding error in the inverse function table (see Sect. 3.1.2). The episode took place on the second day of decryption activities, when the groups had still made only limited use of the representational array, and had yet to discuss or discover a use for the inverse function table. This software bug had managed to elude earlier detection by researchers, and its eventual discovery was precipitated by the events detailed below. The students and the teacher took the repeating letters associated with the bug to be a normal feature of the software, and it was only during the author's review of this episode a few hours later that the phenomenon and the program error behind it were actually identified. Without recognizing the presence of the bug in the software, the group nonetheless engaged it in a process of instrumental genesis, crafting it into a tool and applying it with some success to problem-solving tasks in the *Code Breaker* environment.

4.1.1 Instrumentalization of the Inverse Function Table

The following excerpt finds Jason, in the role of publisher, editing the candidate function as CJ and Vince watched for changes in the other representational artifacts. The students

scrutinized their devices in silence for a few minutes before CJ began to give the following series of reports based on his view of the inverse function table:

CJ: We have two D's, two G's, and two I's, at the moment. Two B's, two E's, two F's, and two J's. Ok, two H's. Three F's. You're getting closer.

Teacher: (passing by the group's table) How do you know you're getting closer?

CJ: Because the letters...update. It's kind of hard to explain. The technology is kind of strange.

Though neither CJ nor the teacher knew it, these multiple appearances of the same letter—"two D's, two G's" and so forth—in the inverse function table resulted from the software's rounding bug. Given that he was simply reading what appeared on the screen, CJ's identification of these repeated letters themselves was unremarkable. But the inference he drew from those repetitions, namely that the group's candidate function was "getting closer" to the solution, reflects the beginnings of a scheme through which he and the group attempted to make strategic use of this accidental software feature.

The teacher overlooked the presence of the bug, but not the significance of CJ's comment. He happened to be passing by the group's table just as CJ told Jason he was "getting closer," and stopped to inquire about the decryption strategy they were using. CJ provided the following explanation:

CJ: I can tell that when two line up and I'll have AA, BB, CC, DD, EE, FF, GG, HH, that's bad. We have A, well, A is alone, and H and I and K and certain letters are alone, which is good. We don't want to have repeats.

Teacher: So you don't want repeats?

CJ: Right.

As Jason, the publisher, adjusted parameters in the candidate function, CJ watched the inverse function table. When he saw two of the same letter "line up" vertically in the table's output column, he inferred that the candidate was "bad." By contrast, if the letters that appeared in the table were "alone," if there were no "repeats," that was "good."

Of course, a tool that simply allowed users to distinguish the single correct encoding function from the infinitely many possible incorrect ones would be of limited code-breaking utility, and other resources in the software already facilitated the same assessment. But CJ's reports to Jason clearly involved a more finely grained distinction than this explanation reveals, namely that some candidate functions were "closer" to the encoding function than others. Vince further elaborated the method in a subsequent conversation with the teacher:

Teacher: So how do you know if it's getting worse or better?

Vince: Well, it, if there's like more doubles, or like triples even, then that's bad. So we go back.

If one repeated letter, or "double," was bad, two were worse, and "triples" of the same letter would be worse still. So as the publisher gradually incremented or decremented a candidate function parameter, the student observing the inverse function table watched for corresponding changes in that display. When repeating letters appeared with increasing frequency, the students inferred that the parameter was being adjusted in the wrong direction, and that they should "go back."

As the group continued to work with the software over the course of the class period, their developing criteria for closeness with regard to their interpretation of the inverse function table became clearer:

CJ: We have a B, C, E, F, H, I, J, K, L, M, N, O, P, Q, R, anyway. That's good because there are more ones.

Vince: So just a couple more.

CJ: That means we're getting close.

In reading off this string of letters, CJ was identifying those which appeared only once in the inverse function table. He reported this information as "good" because it featured "more ones"—more singly appearing letters than previous candidates had yielded. With "a couple more" such letters they would have matched a letter to each distinct numerical character in the encrypted message, and thus presumably broken the code.

These successive articulations of the group's efforts to use the repeated letters in the inverse function table as decoding resources reveal steady refinements in the group's instrumentalization of those accidental artifacts. In his initial description of the approach, CJ characterized the repeating letters as tools for telling "bad" candidate functions from "good" ones. Later, Vince articulated a specific strategy corresponding to these assessments, namely that they should reverse course in the editing of a given candidate function parameter. And finally, CJ described not only the inverse function table states they should edit away from, but also the one they should move toward.

From an instrumental perspective, these inferences took a set of artifacts, the repeated letters, that were aberrant and accidental in the eyes of the designers, and transformed them into meaningful problem-solving tools for the students. The meaning of those tools emerged only through the course of their application to a specific task, namely the group's effort to determine an encoding function. These repeating letters were not intended for such use; they were not intended to exist at all, and their application in this way was certainly not obvious—it did not occur to any of the other student groups participating in the study. The group's interpretation of the repeated letters as problematic was not an entailment of any feature of the inverse function table itself. Rather, it reflects aspects of the ways they perceived the larger code-breaking task, and the nature of the codes as mathematical functions. I will take up these issues in the following section.

4.1.2 Instrumentation of the Inverse Function Table

Whereas the previous section emphasized the ways these learners instrumentalized the rounding bug into a problem solving tool, this section focuses on the ways they instrumented that software artifact—the ways they came to relate the table and the repeated letters to their developing understanding of functions in the *Code Breaker* context. Though "buggy," these repeating letters were certainly not without mathematical significance. By associating two distinct characters in the cipher text with the same plaintext letter, the rounding error gives the impression that the candidate function maps a single input to multiple outputs. Because functions map domain elements uniquely, the presence of "repeats" in the inverse function table would imply that the candidate was not a function at all. The group's descriptions of "repeats," "doubles" and "triples" as "bad," and "ones" or letters appearing "alone" as "good," indicate that their criteria for evaluating candidates related to this property of functions. Vince elaborated this interpretation of the repeating letters while explaining the group's instrumentation of the inverse function table to Reggie, a student in the group who had been absent during the genesis of that instrumentation on the previous day:

There's no such thing as two different numbers for one letter... So, we just, if there's like one A, one B, one C, that's good. If there's like two A's, that's bad. Cause there's only one code for an A...you can't have two codes for one letter.

Vince's rationale for rejecting candidates that produce repeating letters is consistent with a definition of function. He stressed that "there's no such thing as two different numbers for one letter"—two different outputs for one input—because "there's only one code for an A." In other words, the scheme by which the group made use of the rounding bug involved determining whether the mapping of cipher text values to plaintext letters displayed in the inverse function table implied that the candidate was in fact a function, and so possibly the encoding function they sought. As he continued describing the approach to Reggie, Vince modeled the application of such an inference in a decryption task:

Vince: Ok, cause there, for one letter, there's one code, right? Pretend, like, A, it's one. It can't be two or three. So right here it says A, it's negative one...you changed it. Watch. Make it wrong, make it wrong.

Jason: It's already wrong.

Vince: Ok, you see? There's two C's...there's three and five. That's wrong, cause there's only one number for one letter. You get it?

Reggie: Yeah, I get it.

Describing the unique numerical codomain element associated with each letter, Vince explained that if A mapped to one, it could not also map to "two or three" because "for one letter, there's one code." Attempting to show Reggie what he meant as the two boys leaned in to look together at one device, Vince pointed to an A, which the inverse function table showed mapping from negative one only. Surprised because the table had also associated A with additional cipher text values a moment before, Vince complained that Jason had "changed it." Vince instructed Jason to "make it wrong"—to enter a candidate function that would again cause multiple A's to appear in the inverse function table. On Jason's insistence that the candidate currently displayed was "already wrong," Vince scanned further down the table and found that the letter C appeared twice. These repetitions of C were associated with two distinct values in the cipher text, three and five. Consequently, the candidate function was clearly "wrong" because it should show "only one number for one letter."

These excerpts suggest that the group's instrumentation of the rounding error and the resulting repeated letters hinged on the definition of a mathematical function. To implement the bug in the inverse function table as a tool for assessing candidate functions, the students enacted certain assumptions about the relationships between the inputs and outputs of an encoding function represented in the table. These assumptions, such as "there's only one number for one letter," are examples of the *theorems-in-action* guiding this utilization scheme. By creatively drawing on this knowledge of functional relationships, the group managed to transform the repeated letters from the nonsensical artifacts they represented relative to the intended software design into meaningful and useful resources relative to the objective of encoding function identification presented by the decryption task.

4.2 Episode 2: Debugging an Instrument

While the group made admirable progress in instrumenting the rounding bug into a decryption tool over the course of the class session, they never did successfully use the

resulting tool to break a code. The information they gleaned in this fashion helped the group to determine more and less fruitful directions for varying a linear coefficient, but never revealed precise values of any parameters in the encoding function. The next excerpt took place during the same code breaking session as the previous episode, and finds the group again engaging the rounding bug, this time in the context of the graphing feature. While they employed the graphical version of the bug just as they had the tabular one, they also began to make use of other features of the graph in ways that would prove considerably more effective for breaking codes.

4.2.1 Instrumentalization of the Graph

Shortly prior to this episode, CJ had abandoned his view of the inverse function table to study the graph instead. Struck by what he had observed there, he urged his group mates to shift to the graphical view as well:

CJ: You guys scroll up to the...graph. Ok?

Vince: Ok.

CJ: Click on H.

Vince: H.

CJ: See how they're, like, the lines hit it and branched off? That's bad. We want to get rid of that. You need to make that negative number there, your negative twenty-nine higher. And your seventeen lower.

Jason: For the negative, do I make it like a higher negative, or a lower negative? More towards a big negative?

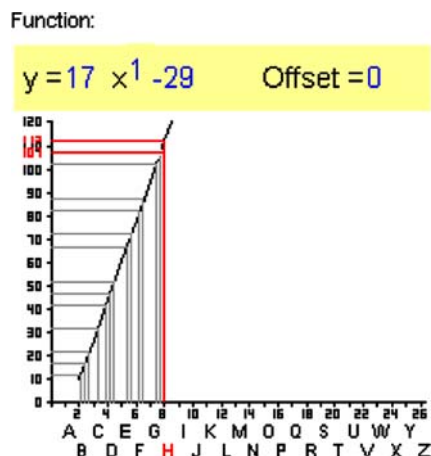
CJ: More towards zero. Make the twenty-nine, start making the twenty-nine more towards zero. And seventeen more towards zero as well.

Vince: Ok, you're getting real close.

CJ: You're getting much closer.

When Vince, at CJ's prompting, tapped his stylus on the letter H that appeared below the number eight on the graph's x -axis, the trace lines associated with H were highlighted in red. As Fig. 9 shows, the vertical red line that extended up from the x -axis to "hit" the candidate function curve "branched off" from this intersection into horizontal red lines

Fig. 9 Graphical manifestation of the rounding bug



leading to two distinct values on the y-axis, 107 and 112, which represented two different characters in the coded text. This phenomenon was a result of the same rounding error that associated multiple numbers with a single letter in the inverse function table, and CJ interpreted it in the same way, as something “bad” that they needed “to get rid of.”

Though he implemented the graphical manifestation of the rounding bug just as he had the tabular version, to reject an incorrect candidate function, CJ also went considerably further here. Rather than simply telling Jason whether he was “getting closer,” CJ this time described much more precisely how to edit the candidate function parameters by making the constant term of “negative twenty-nine higher” and the linear coefficient of “seventeen lower.” Though he did not say as much, he likely did so by drawing inferences from other affordances of the graphing feature as well as those associated with the bug. In the days immediately following this episode, the group would begin making extensive use of the graph as a tool for “fitting” a candidate curve to the viewing window dimensions that had been determined by the encoding function. For example, by examining the graph displayed in Fig. 9, CJ might have deduced that the linear coefficient of 17 made the candidate line too steep to span the full alphabetic domain over the range of cipher text values displayed on the y-axis. Such inferences allowed the group to identify candidate function parameters with considerably greater accuracy in subsequent tasks, and probably this one as well, than had their earlier use of the inverse function table.

I highlight this episode in order to show how the group’s use of the rounding bug in the inverse function table was situated in a larger process of instrumenting the array of artifacts in the handheld software as they gradually developed more effective ways of specifying an encoding function. While they achieved some initial success through their instrumentation of the inverse function table, that success had its limits, and they soon discovered other resources with greater utility for their task. Though the rounding bug and the inverse function table were both removed from the software soon after these events played out, the group appears here to have been on its way to abandoning those artifacts in favor of more useful instruments even prior to that redesign.

4.2.2 Instrumentation of the Graph

In order to elaborate the scheme that appears to have been governing the group’s emerging use of the graph, I examine a subsequent episode in which the group more fully articulated the instrumentation. The following excerpt comes from a code-breaking activity that took place more than a week after the inverse bug discovery:

Vince: How’s that?

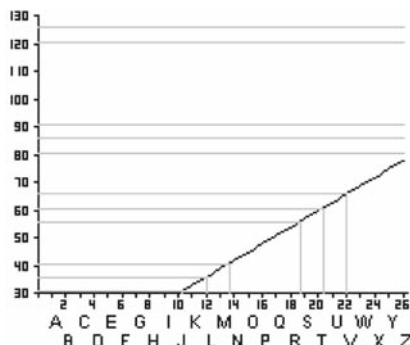
Jason: Terrible. There has to be one for each num... for each letter.

CJ: For each letter in the code.

Figure 10 shows the graphical display Jason and CJ observed as they provided feedback to Vince. Jason asserted that this candidate function was “terrible” because there should be “one” input, represented by a vertical trace line, to complement “each [cipher text] letter” displayed as a horizontal line. CJ’s follow-up clarified Jason’s observation by adding that there needn’t necessarily be 26 different vertical trace lines, but rather just enough to match the 11 distinct letters in the encrypted message. Together, they recognized the need for one input value corresponding to each cipher text letter-output.

Jason articulated this reasoning more explicitly during a subsequent interview with a researcher:

Fig. 10 Graph of a “terrible” candidate function



the graph, it wasn't really...complete. It was only a little bit of the way up, so the top lines...they're like just straight across, and they weren't really assigned to any letters or anything, so that meant that...the coefficient had to be higher, so it completed the graph.

Jason's explanation emphasizes two important and related aspects of this utilization scheme: that the graph of the candidate function should “complete” the viewing window, and that it should “assign” each horizontal trace line and its corresponding cipher text value to a plaintext letter. In other words, the group used both the software's scaling features and the trace lines to establish boundaries within which they might fit a candidate curve. In doing so, they relied on the one-to-one correspondence between the input and output values of an encoding function as a resource for evaluating each candidate. In Episode 1, the group was making use of the functional status of substitution codes, as requiring “*only one*” cipher text character for one plaintext letter, to instrument the inverse function table bug. Similarly, this episode demonstrates how their instrumentation of the graph relied on the surjectivity of those substitution codes, such that “*there has to be one*” input for each output.

4.3 Episode 3: Redesigning an Artifact

On discovering the rounding bug, the author and other research team members were faced with a dilemma: to rapidly revise the software at the risk of disrupting implementation of the instructional unit, or to leave the bug in place and risk the learner confusion and misconceptions to which it might contribute. Pedagogical concerns soon tilted the balance in favor of revision; initial observations of inverse function table use across many groups suggested that students examining this representation tended to focus on the plaintext letters, repeating or not, and ignored the numerical cipher text values and the possible functional relationships that produced them. These tendencies may have reflected the table's pairing of a dynamic letter display with a static display of the numerical outputs; as the elements of the representation that changed with adjustments to the candidate function parameters, the letters were natural objects of interest. The resolution involved eliminating both the buggy rounding feature and the inverse function table altogether, and replacing the latter with a function table (see Sect. 3.1.4). Because this new representation paired static alphabetic inputs with dynamic candidate outputs, we hoped it would more effectively support both students' decryption efforts and our learning goals. The episode that follows

took place after this change in the software, and captures the efforts of Vince and Jason to shape this new representational artifact into a code-breaking tool.

4.3.1 Instrumentalization of the Function Table

In the following excerpt, the group had already been working for several minutes on this code, and had used the graph and other resources to deduce a cubic candidate function with a lead coefficient in the high teens. Upon changing the lead coefficient to a value of 18, Vince asked Jason whether the values in the resulting function table were approaching those of the frequency table:

Vince: OK. Tell me if I'm close. How's that?

Jason: Yeah, it's close.

Vince: Do I have any...

Jason: Yeah, you have a match, right here [points to the frequency table on his PDA].

Vince: I do? One?

Jason: Yeah.

Vince: Ok. How about now?

Jason: Wait. Um, it's a little bit off...Keep on going... Off, down. We're pretty close. [Jason talks off-topic with another student for a moment while Vince adjusts his view to compare the tables himself]

Vince: Ok. Our lowest number is 24, then the real lowest number is...one hundred...eh, no, the real lowest number is 20. I'm going... OK, we got it.

First confirming that the candidate function was “close,” Jason then reported that he had found “a match,” a pairing of identical, rather than merely similar, values in the function and frequency tables. Vince confirmed that there was only “one” such match, and then returned to editing the candidate by adjusting the coefficient to 19. Jason continued to provide feedback based on the tables as Vince sought an appropriate constant value for the new coefficient. Jason noted that an unspecified function table value was “a little bit off” its counterpart in the frequency table, and that Vince should “keep on” adjusting the candidate “down.” Upon observing the tables himself, Vince compared a low value of 24 in the function table with a “real” low of 20 in the frequency table. He then reduced the constant from five to one, announcing that “we got it” as the correct plaintext message was revealed.

In the first episode, the group's use of the inverse function table focused on scrutinizing that representation to identify problematic features associated with a candidate function, and then editing away from that candidate so as to eliminate those features. By contrast, their technique here involves directly comparing candidate output values in the function table with the cipher text values displayed in the frequency table in order to edit the candidate *toward* the encoding function. The students sought potential “matches” between values in the two tables, checked to see if a given candidate produced only one such match or many, and edited the candidate function to bring near-matches into exact alignment. Importantly, the group's efforts to decode the message in the above excerpt also relied heavily on the instrumentalization of the graph described in the previous episode. Inferences from the window dimensions and trace lines had allowed Vince to determine the correct exponent and approximate the coefficient prior to his and Jason's efforts here to fine-tune the latter parameter and specify the constant using the frequency and function tables.

4.3.2 Instrumentation of the Function Table

In comparing values in the frequency and function tables, Vince and Jason focused closely on the relationship between candidate function parameters and numerical output values. In doing so, they demonstrated two important aspects of the group's utilization scheme for this instrument. First, they distinguished between candidates that matched one and multiple function table values with the array of cipher text values in the encrypted message. Vince's rejection of the candidate function with only one match follows from the inability of a single ordered pair to uniquely determine the encoding function. Moreover, his choice to edit the coefficient rather than the constant reflects the fact that any translation of the candidate would only achieve a new match at the expense of the current one.

When that change in the coefficient produced an array of near-matches, Vince focused on one pair of those "close" values and adjusted the constant until they were identical. This act highlights a second key feature of the group's utilization scheme. Rather than directly calculating the output values to which a candidate function mapped various inputs, Vince relied on the capacity of the function table to compute and display those outputs automatically. In reducing the constant of five to a one to produce a corresponding reduction in the output, he used the parameter as a tool for translating the set of candidate function outputs in relation to the cipher text values. In other words, the scheme through which these students utilized the function table treated changes in the candidate function not simply in terms of operations on a single input value, but as transformations of a set of output values.

4.4 Discussion

Taken together, these episodes highlight some of the ways the theory of instrumental genesis can serve as a resource both for accounting for unexpected learning outcomes—here made particularly salient by their reliance on an unintended tool feature—as emergent consequences of mediated activity, and for examining the dialectic between goal-driven, tool- and task-specific processes, and complex concepts like mathematical functions. At the same time, they also illustrate the emergent interplay between learners' tuning of an instrument toward a problem-solving task, and designers' tuning of an artifact toward a set of learning objectives underlying both task and tool design. In this section, I summarize and reflect on lessons from these three episodes, first from the standpoint of instrumental genesis, and then in terms of the emergent design of a learning environment.

4.4.1 Decryption Instruments and Function Characteristics

The dual processes of instrumentalization and instrumentation provide a framework for simultaneously examining both the successive techniques through which students apply artifacts to achieve problem-solving goals, and the ways they engage curricular objectives through those techniques. Table 1 summarizes key features of the group's engagement with the decryption tasks, and with aspects of the function concept, as they worked with different representational elements of the *Code Breaker* software over the three episodes.

Over the course of the instructional unit, these students worked together to develop and refine strategies for solving an array of increasingly difficult function-based decryption

Table 1 *Code Breaker* instruments and aspects of function

Representational artifact	Inverse function table (Episode one)	Graph (Episode two)	Function table (Episode three)
Instrumentalization	Look for “repeats” in the inverse function table	Look for letters that “branch off” to multiple numbers	Identify similar values in the frequency and function tables
	Edit candidate function parameters in the direction that produces fewer repeats and more “ones”	Edit the candidate to: – fit graph to a viewing window determined by the encoding function – connect horizontal and vertical “trace lines” in the graph	Determine whether one or multiple values “match” Edit exponent and coefficient to generate matches, edit constant to translate nearly matching values
Instrumentation	Emphasizes the unique mapping from plaintext to cipher text elements	Emphasizes the one-to-one mapping between plaintext and cipher text elements	Multiple ordered pairs required to uniquely determine an encoding function Candidate parameters as transformations of a set of output values

tasks using the *Code Breaker* software on their networked devices. The three episodes above illustrate ways those strategies used various representational artifacts provided by the handheld software to match a candidate function with an unknown encoding function. Through microgenetic processes only partly elaborated in the accounts reported here, these artifacts emerged as three quite distinct instruments in the group’s decryption activity. These instruments reflected not only very different ways of applying the software tools to the code-breaking tasks, but also very different ways of engaging the function concept. In episode one, the group’s instrumentation of the inverse function table depended on a defining characteristic of mathematical functions, as unique mappings from each input value to an associated output value. That property of encoding functions was made salient by a distinctive feature of the inverse function table, namely the “buggy” tendency to associate multiple outputs with a single input for certain incorrect candidate functions. Similarly, in episode two the group instrumented the graph through a scheme emphasizing the one-to-one correspondences between plain and cipher text elements specified by an encoding function. Once again, this instrumentation emerged in concert with the instrumentalization of characteristics of the *Code Breaker* graphical interface that reflected this one-to-one correspondence, namely the dimensions of the viewing window and the trace line feature. And finally, in episode three the group instrumentalized the function table’s capacity to automatically compute and display the complete set of outputs to which a given candidate mapped the alphabetic inputs by coupling this dynamic representation with a static array of cipher text values in the frequency table. The accompanying instrumentation followed both from the multiple ordered pairs necessary to uniquely specify a polynomial function, and from the links between adjustments to candidate function parameters and transformations of the set of function table outputs.

Each of these instruments reflects emergent properties of the intersection among students, software constraints, and task components. For example, the group’s developing utilization schemes for the inverse and function tables and the graph all reflected ways

those representations interacted with unique aspects of the candidate function interface through the process of seeking an encoding function. Because candidate parameters must be unitarily incremented or decremented from their current values rather than entered directly, the process of editing a parameter often made patterns associated with those gradual changes, such as the steady increase in a slope or translation of a graph, particularly salient. In the case of episode one, the mere appearance of repeated letters in the inverse function table likely would not have supported the students' inferences regarding a direction in which to edit the candidate. But the appearance of more or fewer such repeats as they adjusted candidate parameters, coupled with their recognition of those repeats as inappropriate to the mathematical context, provided a backdrop against which the group interpreted the changes relative to the problem-solving task. Similarly for the second episode, the instrumental meaning of the candidate graph reflected the process through which the editing of parameters led toward or away from a "complete" fit between curve and viewing window. And in episode three, changes to candidate parameters became meaningful in simultaneous relation to both transformed function table values and a static frequency table display. The ideas about function associated with these various schemes thus each emerged through a particular interplay of artifact and goal-oriented activity, literally as theorems-in-action, related to those manifestations of the candidate and encoding functions provided in each representation.

Some of these instruments achieved the students' problem-solving objectives more effectively than others. While their use of the inverse function table never led to the successful decryption of a message, the group drew heavily on the graph in breaking several codes throughout the unit. And their instrumentalization of the function table was the critical step that allowed them to solve the most challenging codes they broke. Moreover, their instrumentation of the function table was arguably the richest among those presented here with regard to the breadth and depth of understanding about functions and their representations that it demonstrated. Of the three instruments summarized in Table 1, only that involving the function table featured students' examination of candidate function parameters in relation to specific numerical output values. And only that instrumentation highlighted changes to those parameters in terms of transformations on those output values, or the multiple ordered pairs required to uniquely specify a function. Finally, while the other instruments involved coordinating the algebraic candidate function expression with one other representation, this one depended on the simultaneous use of two other representations, in the form of the function and frequency tables. So in addition to greater efficacy as a tool for solving, the redesigned software with the function rather than inverse function table appears to have provided a more effective tool for learning. The next section explores this point in greater detail, and from the standpoint of the emergent design of the *Code Breaker* tools.

4.4.2 Emergent Design: Aligning Users' and Designer's Objectives

From another perspective, these three episodes detail the emergent design of a learning environment. The first episode helped bring to light both a technical and a conceptual flaw in the *Code Breaker* design, the former in the form of the bug and the latter in terms of the match between the inverse function table and the objectives for student learning about functions. Each of the *Code Breaker* representations was intended to make certain aspects of function particularly salient to learners; together, this set of representations and the collaborative decryption tasks were intended to emphasize connections among represen-

tational modes and to broadly support students' developing understanding of functions. Within this larger set of student learning objectives, the inverse function table was designed to emphasize the analogy between decrypting a message and inverting or undoing a function. Moreover, the table included a rounding feature intended to make the representation more useful in decryption tasks. Ironically, though, the very utility of this feature with or without its bug appeared to undermine rather than achieve the learning objective behind the design; students focused on whether, which and how many letters appeared in the table rather than on the relationship between an algebraic candidate function and any particular numerical mapping from output to input. In fact, the instrumentation of the rounding bug into a tool for identifying unique function mappings by the group described here was the most successful use, with regard to both the problem-solving and instructional objectives, of the inverse function table made by any of the groups in the class. And in that case the desired student learning, in the form of a utilization scheme based on aspects of function, emerged only when the table, because of the bug, failed to correctly depict the inverse relationship it was devised to illustrate.

This failure demonstrates the pivotal relationship between utility and instructional efficacy, or between task-oriented and learner-oriented activity, at the heart of the dialectic of instrumental genesis and design research. Tools for learning achieve their educational objectives when their instrumentalization—their orientation by the user-learner toward a task—invites an instrumentation that incorporates those objectives. The inverse function table may have provided a rich representation of decryption as undoing from the standpoint of designers acquainted with those aspects of function, but it did not provide task-relevant utility that made that acquaintance necessary or those aspects salient to learners. At best, and thanks to a bug, it supported an instrumentation that featured an important but partial account of functional relationships—specifically, that instrumentation emphasized the unique mapping from input to output, but not the reciprocity with inverse mappings intended by the designers. So the group's instrumentalization of the bug helped to reveal ways the design of the inverse function table constrained the possibilities for its instrumentation, and thus led to its replacement with the function table.

This new design abandoned a focus on inverse in order to capitalize on the students' demonstrated tendency to draw inferences from multiple simultaneously displayed table values. Both the function and inverse function tables were intended to be read horizontally, to emphasize the between-variable relationships illustrated across the rows of each table. The students' instrumentalization of the inverse function table instead relied on reading vertically, focusing on the within-variable relationships of the letter column. While the table also in principle afforded a meaningful horizontal reading, this vertical emphasis emerged as a consequence of the dynamic relationship between candidate function parameter change, letter displays, and task objective. By instead making dynamic the column displaying a set of numerical output values, the function table capitalized on a similarly emergent relationship between tool and task. Now, however, students' vertical reading of the function table combined with a vertical reading of output values in the frequency table—a between-columns comparison that better supported both learners' and designers' objectives. In other words, the new table reflected a revised orientation toward the learner that took into account the learner's orientation of the previous artifact toward the decryption task.

Of course, the students showed signs of anticipating that design shift in episode two, when they began favoring the graph over the inverse function table as a decryption resource. Because they found ways of using the graph that allowed them to converge on an accurate approximation of the candidate function more efficiently and effectively than they had with

the inverse function table, they had already begun using the graph more and the table less when the latter representation was replaced in the software. In other words, the group began to debug an instrument even before the designers debugged the corresponding artifact, abandoning the inverse function table in search of better ways of applying the array of *Code Breaker* representational tools to the decryption tasks. Thus, while the group re-instrumentalized that set of artifacts in search of better tools for breaking codes, the researchers redesigned the software in search of better tools for supporting student learning.

5 Conclusion

This article recounts the discovery by a group of students of a software bug in a novel learning environment. The episode occasions the unfolding of two distinct but overlapping stories. The first of these tales features the group's surprisingly successful efforts to fashion the bug into a meaningful mathematical instrument in the context of an assigned problem-solving task, and their subsequent abandonment of that instrumented bug in favor of another tool with greater efficacy relative to that task. The second focuses on the designers' decision to replace the buggy feature with a different artifact, and the group's instrumentation of that new tool as it emerged relative to the learning objectives of the problem-solving activity and environment. Each of these stories begins with the appearance of the bug, and ends with the abandonment of the buggy artifact, though the respective narrative journeys are quite different. Together, the two vantage points from which these events are recounted suggest a way of characterizing the emergence of classroom tools as resulting from dialectical interactions between processes of design and instrumentation. In other words, as a set of software artifacts was designed and redesigned by researchers, learners crafted a succession of mathematical instruments from those artifacts. And through observation and analysis of the ways learners shaped problem-solving tools from those successive artifacts, researchers gleaned insights to guide their redesign efforts, both during and after that round of implementation.

Conceptualizing the design and investigation of innovative learning tools in terms of this dialectic has the potential for contributing both to the theory of instrumental genesis and the epistemology of design-based research. While artifacts in the theory of instrumental genesis tend to appear as black boxes, design research requires continually reopening those boxes and reevaluating the features and constraints of an artifact in the light of new empirical insights. Focusing on design thus adds another productive set of tensions to the instrumental dialectic—namely between the educator-designers of classroom tools and the learner-users of those tools. At the same time, the instrumental theory provides a potentially compelling resource to design researchers, particularly in the form of the links between instrumentation and instrumentalization. This framework is powerful precisely because it integrally intertwines emerging conceptual understanding with a learner's goal-oriented tool use. In the case of the rounding bug, for example, attending to the ways the focus group applied the repeating letters to the decryption task provided a window into the group's emerging understanding of function, particularly as that application emphasized key aspects of functional relationships other than the learning goals about inverse mapping specified by the designers. And replacing the inverse function table with the function table reflected a change in the designers' orientation of the artifact toward the learner in order to account for the latter's orientation of that artifact toward the task, such that the learner's instrumented activity and the designers' overarching goals for student understanding of function were brought into closer alignment.

The episode involving the discovery of the bug demonstrates this compatibility between the instrumental genesis and design research perspectives by lending particular salience to aspects of both. The group's embrace of this accidental software feature emphasizes the nature of tool use as emergent from features of an instrumented activity system rather than predetermined by design intentions. The point is not so much that the software included a bug as that the presence of that bug made particularly visible the capacity of these users to make unexpected use of features of the artifact, even features the designers did not know were there or intend to include. Instrumental genesis always includes the potential for unexpected forms of tool use, but the unexpected use of an unknown bug makes that potential particularly visible in this episode. Indeed, instrumental theory not only offers a conceptual framework for interpreting the mathematical meaning students made from this bug, but also explains its serendipitous discovery. By characterizing tools as emerging not only from a design, but also from negotiations between particular users and tasks, the instrumental genesis perspective provides a way to account for and analyze the novel and unexpected ways students might make use of classroom artifacts.

This article details the *instrumentalization-in-process* of an *artifact-in-progress*, telling a story about the simultaneous fluidity of designed artifacts, as subject to successive cycles of debugging and redesign, and instruments, as subject to successive cycles of genesis. Admittedly, most classroom artifacts are far less fluid, and new instruments can emerge at most only as frequently as students are provided with either new tools, or new and sufficiently rich tasks to require applying familiar devices in new ways. But instrumental geneses should be seen as learning opportunities for designers and other educators as much as for students. The capacities of the students and designers described here to learn from working with and working past the flaws in a tool constitute valuable resources for processes of design and instruction alike. Yerushalmy's distinction between tools for solving and for exploring may point to a powerful way for capitalizing on those resources. The *Code Breaker* environment represents an attempt to provide both—presented with a set of artifacts for solving open-ended problems, students must explore strategies for making those artifacts into meaningful and effective tools. As a first iteration of that design, the environment certainly requires further revision, followed by subsequent rounds of investigation into the forms and the degree of support for mathematical learning; this work is underway. But the version presented here, bugs and all, speaks to the potential utility for designers, and the surprising mathematical richness, of learners' unscripted tool use. Such instances of instrumental genesis might be productively seen not as disruptions, but as objectives in the process of designing tools for learning.

Acknowledgements Support for this project was provided by Stanford University, The Wallenberg Global Learning Network, and Hewlett-Packard Corporation. Thanks to Roy Pea and Shelley Goldman for shepherding the project, and Jo Boaler the dissertation, from which this article emerged. I'm also deeply indebted to Mike Stieff, Richard Noss, and three anonymous reviewers for their insightful comments on earlier drafts.

References

- Artigue, M. (2002). Learning mathematics in a CAS environment: The genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. *International Journal of Computers for Mathematical Learning*, 7(3), 245–274.
- Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher*, 32(1), 9–13.
- Cole, M. (1996). *Cultural psychology: A once and future discipline*. Cambridge: Harvard University Press.

- Cole, M., & Engestrom, Y. (1993). A cultural-historical approach to distributed cognition. In Salomon (Ed.), *Distributed cognitions: Psychological and educational considerations* (pp. 1–46). New York: Cambridge University Press.
- Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: Theoretical and methodological issues. *The Journal of the Learning Sciences*, 13(1), 15–42.
- Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, 32(1), 5–8.
- diSessa, A., & Cobb, P. (2004). Ontological innovation and the role of theory in design experiments. *The Journal of the Learning Sciences*, 13(1), 77–103.
- Guin, D., & Trouche, L. (1999). The complex process of converting tools into mathematical instruments: The case of calculators. *International Journal of Computers for Mathematical Learning*, 3(3), 195–227.
- Haspekian, M. (2005). An “instrumental approach” to study the integration of a computer tool into mathematics teaching: The case of spreadsheets. *International Journal of Computers for Mathematical Learning*, 10(2), 109–141.
- Hegedus, S. (2005). Dynamic representations: A new perspective on instrumental genesis. In *Proceedings of the fourth congress of the European society for research in mathematics education*.
- Lagrange, J. B. (1999). Complex calculators in the classroom: Theoretical and practical reflections on teaching pre-calculus. *International Journal of Computers for Mathematical Learning*, 4(1), 51–81.
- Mariotti, M. (2002). The influence of technological advances on students’ mathematics learning. In L. English (Ed.), *Handbook of international research on mathematics education* (pp. 695–723). NJ: Lawrence Erlbaum Associates.
- Mehan, H. (1989). Microcomputers in classrooms: Educational technology or social practice? *Anthropology and Education Quarterly*, 20(1), 4–22.
- Meira, L. (1998). Making sense of instructional devices: The emergence of transparency in mathematical activity. *Journal for Research in Mathematics Education*, 29(2), 121–142.
- Orlikowski, W. (1992). The duality of technology: Rethinking the concept of technology in organizations. *Organization Science*, 3(3), 398–427.
- Ruthven, K. (2002). Instrumenting mathematical activity: Reflections on key studies of the educational use of computer algebra systems. *International Journal of Computers for Mathematical Learning*, 7(3), 275–291.
- Suchmann, L. (1986). *Plans and situated actions: The problem of human-machine communication*. New York: Cambridge University Press.
- Suchmann, L., Blomberg, J., Orr, J., & Trigg, R. (1999). Reconstructing technologies as social practice. *American Behavioral Scientist*, 43(3), 392–408.
- Trouche, L. (2004). Managing the complexity of human/machine interactions in computerized learning environments: Guiding students’ command process through instrumental orchestrations. *International Journal of Computers for Mathematical Learning*, 9(3), 281–307.
- Verillon, P., & Rabardel, P. (1995). Cognition and artifact: A contribution to the study of thought in relation to instrumented activity. *European Journal of Psychology in Education*, 9(3), 77–101.
- Vygotsky, L. (1978). *Mind in society*. Cambridge, MA: Harvard University Press.
- White, T. (2006). Code talk: Student discourse and participation with networked handhelds. *International Journal of Computer-Supported Collaborative Learning*, 1(3), 359–382.
- Yerushalmy, M. (1999). Making exploration visible: On software design and school algebra curriculum. *International Journal of Computers for Mathematical Learning*, 4(2–3), 169–189.