software studies\ a lexicon

edited by **matthew fuller**

SOFTWARE STUDIES

Software Studies

A Lexicon

edited by Matthew Fuller

The MIT Press
Cambridge, Massachusetts
London, England

© 2008 Matthew Fuller

Individual texts © copyright of the authors, 2006

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

For information about special quantity discounts, please email special_sales@mitpress.mit.edu

This book was set in Garamond 3 and Bell Gothic by Graphic Composition, Inc.

Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data
Software studies : a lexicon / edited by Matthew Fuller.
p. cm.—(Leonardo books)
Includes bibliographical references and index.
ISBN 978-0-262-06274-9 (hbk. : alk. paper) 1. Computer software. 2. Computers and civilization—Encyclopedias. 3. Programming languages (Electronic computers)—Lexicography. 4. Technology and the arts. I. Fuller, Matthew.
QA76.754.S64723 2008
005.1—dc22

2007039724

10 9 8 7 6 5 4 3 2 1

Contents

SERIES FOREWORD ix
ACKNOWLEDGMENTS xi

INTRODUCTION
Matthew Fuller 1

ALGORITHM
Andrew Goffey 15

ANALOG
Derek Robinson 21

BUTTON
Søren Pold 31

CLASS LIBRARY
Graham Harwood 37

CODE
Friedrich Kittler 40

CODECS
Adrian Mackenzie 48

COMPUTING POWER
Ron Eglash 55

CONCURRENT VERSIONS SYSTEM
Simon Yuill 64

COPY
Jussi Parikka 70

DATA VISUALIZATION
Richard Wright 78

ELEGANCE
Matthew Fuller 87

ETHNOCOMPUTING
Matti Tedre and Ron Eglash 92

FUNCTION
Derek Robinson 101

GLITCH
Olga Goriunova and Alexei Shulgin 110

IMPORT/EXPORT
Lev Manovich 119

INFORMATION
Ted Byfield 125

INTELLIGENCE
Andrew Goffey 132

INTERACTION
Michael Murtaugh 143

INTERFACE
Florian Cramer and Matthew Fuller 149

INTERNATIONALIZATION
Adrian Mackenzie 153

INTERRUPT
Simon Yuill 161

LANGUAGE
Florian Cramer 168

LISTS
Alison Adam 174

LOOP
Wilfried Hou Je Bek 179

MEMORY
Warren Sack 184

OBFUSCATED CODE
Nick Montfort 193

OBJECT ORIENTATION
Cecile Crutzen and Erna Kotkamp 200

PERL
Geoff Cox and Adrian Ward 207

PIXEL
Graham Harwood 213

PREFERENCES
Søren Pold 218

PROGRAMMABILITY
Wendy Hui Kyong Chun 224

SONIC ALGORITHM
Steve Goodman 229

SOURCE CODE
Joasia Krysa and Grzesiek Sedek 236

SYSTEM EVENT SOUNDS
Morten Breinbjerg 243

TEXT VIRUS
Marco Deseriis 250

TIMELINE (SONIC)
Steve Goodman 256

VARIABLE
Derek Robinson 260

WEIRD LANGUAGES
Michael Mateas 267

BIBLIOGRAPHY 277
ABOUT THE CONTRIBUTORS 313
INDEX 321

Interaction

Michael Murtaugh

Types of interaction can be categorized in a variety of ways. One popular sort of interaction consists of the user making choices, either textual or via a graphical user interface: selecting items from a menu; typing bits of information into a form; moving a mouse; clicking; double-clicking. The popularity of the web and hypertext has bound the idea of interaction to branching link structures. The word "interactive" has become so overused in relation to computing and new media that, for instance, Lev Manovich describes its use in relation to computing as "tautological" and takes care to qualify any employment of the word when unable to avoid it altogether.[1]

Interaction is also linked to a tradition of engineers, mathematicians, and software hackers looking for a way to break out of the rigidity and the strictness of their systems—out, as it were, of the black box. Interaction in the 1960s represented reaction against, and liberation from, the mainframe batch-processing computer center. It proposed a radical usage of computers: giving (untrained) groups of users "live" contact with the machine.

Input Tape

An early proponent of this new approach to computation was J. C. R. Licklider, a researcher with a background in psychoacoustics. In the 1950s Licklider had access to the TX-2, an experimental computer developed at MIT that, along with having a Cathode Ray Tube display, speaker, control knobs, and a light pen, was fully transistor-based. The TX-2 could be readily reprogrammed from its keyboard instead of requiring physical rewiring or the use of punchcards.[2]

It was on the TX-2 that Ivan Sutherland would later develop his Sketchpad program, cited by many, including Licklider, as a groundbreaking demonstration of the potential for truly interactive software.[3] Licklider describes the then "state of the art":

> Present-day computers are designed primarily to solve preformulated problems or to process data according to predetermined procedures. The course of the computation may be conditional upon results obtained during the computation, but all the alternatives must be foreseen in advance. If an unforeseen alternative arises, the whole

process comes to a halt and awaits the necessary extension of the program. . . . If the user can think his problem through in advance, symbiotic association with a computing machine is not necessary. . . One of the main aims of man-computer symbiosis is to bring the computing machine effectively into the formulative parts of technical problems.[4]

Licklider links interaction to a crucial shift from computer as problem-solver to computer as problem-finder or problem-explorer in a space of necessarily unforeseen possibilities.

Writing in the 1980s, cybernetician Stafford Beer describes an algorithm as "a technique, or mechanism, which prescribes how to reach a fully specified goal." He contrasts this with the idea of a heuristic (method), a word derived from the adjective meaning "to find out":

An heuristic specifies a method of behaving which will tend towards a goal which cannot be precisely specified because we know what it is but not where it is. . . . The strange thing is that we tend to live our lives by heuristics, and try and control them by algorithms. Our general endeavour is to survive, yet we specify in detail ("catch the 8.45 train," "ask for a raise") how to get to this unspecified and unspecifiable goal. We certainly need these algorithms, in order to live coherently; but we also need heuristics—and we are rarely conscious of them.[5]

Writing in the 1990s, computer science theorists Peter Wegner and Dina Goldin provide another description of an algorithm: "A systematic procedure that produces—in a finite number of steps—the answer to a question or the solution to a problem.[6]

Wegner and Goldin propose an alternative to the Turing Machine based around a unifying concept of interaction. In the classic formulation, a Turing machine is an idealized computer that reads and writes symbols from an endless tape and has a notion of "state," allowing a program written in these symbols to control the operation of the machine. The rules for this particular model dictate that once the machine begins operation, no new input may be received, and it must be guaranteed to reach a final state in a fixed amount of time. In addition, the starting state for the machine is precisely specified and must be identical each time the machine is started.

In contrast, Wegner and Goldin propose "interactive computation" based on "interaction with an external world, or the environment of the computa-

tion, during the computation—rather than before and after it, as in algorithmic computation.[7]

In one alternative model called a Persistent Turing Machine, Wegner and Goldin describe a variation on the classic Turing machine, now with tapes. The crucial differences in this model are: (1) the use of an input and an output tape to interface with the dynamic environment of the machine, (2) the provision for a work tape that remembers results from previous operation, and (3) the allowance for the machine to run continuously (no requirement to reach a final state). By writing to and subsequently reading from the environment, the potential for feedback occurs. It is, however, a noisy channel, as the environment is explicitly allowed to be unpredictable, and potentially acted upon by other processes. Allowing past operation to influence the starting state also introduces a greater degree of uncertainty. Allowing indefinite operation reflects a more heuristic-driven approach as a result may be "tended towards" without necessarily being definitively reached.

By explicitly representing a place "outside" of the machine, Wegner and Goldin show that the resulting model is more expressive, able to describe machines that are not possible to fully represent as traditional Turing machines. In addition, they show how such a model fits much more readily with the realities of contemporary computation such as operating systems, networked software, and portable devices.[8]

Wegner and Goldin point out that Turing himself acknowledged other potential models that might include human interaction (choice-machines) or other external inputs (oracle-machines).[9]

Talking to to Stuart Brand in the 1980s, Andy Lippmann, director of an early experimental videodisc called the *Aspen Movie Map*, provided the following working definition of interactivity: "Mutual and simultaneous activity on the part of both participants usually working toward some goal, but not necessarily."[10]

The *Aspen Movie Map* was an attempt to recreate the experience of exploring a city by virtually driving through the city's streets, selecting points of interest along the way to view in depth. The challenge for Lippman and the other designers of the project was to realize that goal within the extreme limitations of a static and limited storage medium.

Lippman describes five "corollaries" or properties he felt were necessary to add to this to attain true interactivity: interruptibility; graceful degradation; limited look-ahead (not pre-computed); no default pathway, and; the "impression of an infinite database."[11]

Work Tape

Surveying these different perspectives, some themes seem to emerge as central to interaction in relation to computation: liveness, plasticity and accretion, interruption and incompleteness.

Liveness

A key recurrent theme in interactivity is liveness. Licklider states that a central aim for human-computer symbiosis is "to bring computing machines effectively into processes of thinking that must go on in 'real time.'" Lippman's "limited look ahead," the importance of computational decisions happening "on the fly" is paralleled by Wegner and Goldin's notion of noncomputability, the idea that not all possible pathways can be precomputed.

The idea of the "infinite database" is the subjective counterpoint to this noncomputability and liveness of the system: the feeling that there are infinite possibilities to explore. A result of this liveness is that an interactive system is one that supports a sense of playing or performing with the system.

An important consequence of liveness is that interaction always occurs over time. In black box computation, time is neglible—the only requirement is that computation completes in a finite time. There is no sense of the "mutual" or simultaneous in algorithmic computation, all computation is completed before any results are passed on to the next process.

Interaction always involves simultaneity, as computation occurs iteratively through feedback to a shared and changing environment. Designing with interaction requires a sensitivity to the timing of the processes involved.

Plasticity and Accretion

Licklider, formulating the idea of the computer as a communication device uses the term "cooperative modeling," writing, "Creative, interactive communication requires a plastic or moldable medium that can be modeled, a dynamic medium in which premises will flow into consequences, and above all a common medium that can be contributed to and experimented with by all."[12]

Lippmann's notions of interruptibility and graceful degradation express the desire for a kind of plasticity to the experience; the participant can push without breaking the system.

The idea of malleability and plasticity connects back to the central role of persistence in Wegner and Goldin's interactive computation. The fact that

the computation "holds its shape" in some sense requires that the interaction have some lasting effect (if only short-term). Interaction includes the potential for processes to improve or evolve with accretion. The lack of a fixed starting state or default pathway underscores the importance of accretion to the computation.

Interruption and Incompleteness

For Lippman, it is the potential for interruption that keeps a conversation from becoming merely a lecture. In Wegner and Goldin's interactive computation, it is in the noisy channel of the environment that interuption potentially occurs. The environment of the computation serves as the interface between the various processes, be they purely computational or the result of human intervention.

The desire for graceful degradation relates to the idea that the computation must not only be open to unpredictable input, but should use it well.

Models of the real world and even of integers sacrifice completeness in order to express autonomous (external) meanings. Incompleteness is a necessary price to pay for modeling independent domains of discourse whose semantic properties are richer than the syntactic notation by which they are modeled.[13]

The components of an interactive system are inherently incomplete. Interaction always involves a tension between autonomous operation and cross-influence of a system's parts. The challenge for authors is to design processes that tend to steer the system toward desirable states rather than hard coding those states. An interactive process exploits its environment in order to fully realize its own functionality.

Output Tape

For the software designer, programming with interaction involves seeking a kind of magical moment of transformation, a moment when one begins to get back more than what was put in; an unexpected moment when the system seems not only just to work, but to almost come to life; a moment when what had previously been a noisy mess of buggy half-working mechanisms seems to flow together and become a kind of organic whole.

The elusive chase for this kind of transformative moment is the essential reason why geeks keep banging away at their keyboards, deep into the night, deprived of sleep and propped up by caffeine and sugar and the adrenaline of the experience of feeling in contact with something larger than oneself.

Interaction rips computation out of the clean room of the algorithm and thrusts it into the tainted and unpredictable space of dynamic and shared environment.

Interaction forces a rethinking of algorithmic approaches toward those that perform a kind of dance alternating between active computation and responsive strategies to a changing environment.

Embracing interaction requires the programmer or designer to break open the black boxes of algorithmic processes and acknowledge the incompleteness of what they create in the pursuit of experiences that are playful, insightful, and potentially surprising.

Notes

1. Lev Manovich, *The Language of New Media*, 55.

2. M. Mitchell Waldrop, *The Dream Machine: J. C. R. Licklider and the Revolution that Made Computing Personal*, 142–147.

3. Ibid, 255.

4. J. C. R. Licklider, "Man-Computer Symbiosis," *IRE Transactions on Human Factors in Electronics, vol. HFE-1*, 4–11.

5. Stafford Beer, *Brain of the Firm*, 52–53.

6. Eugene Eberbach, Dina Goldin, and Peter Wegner, "Turing's Ideas and Models of Computation" 1 online version, available at http://www.cis.umassd.edu/~eeberbach/papers/eberbach_12092003.pdf.

7. Ibid, 16.

8. Peter Wegner, "The Paradigm Shift from Algorithms to Interaction: Why Interaction is More Powerful than Algorithms," *Communications of the ACM*.

9. Eberbach, Goldin, and Wegner, "Turing's Ideas and Models of Computation."

10. Stuart Brand, *The Media Lab*, 46–50.

11. Ibid.