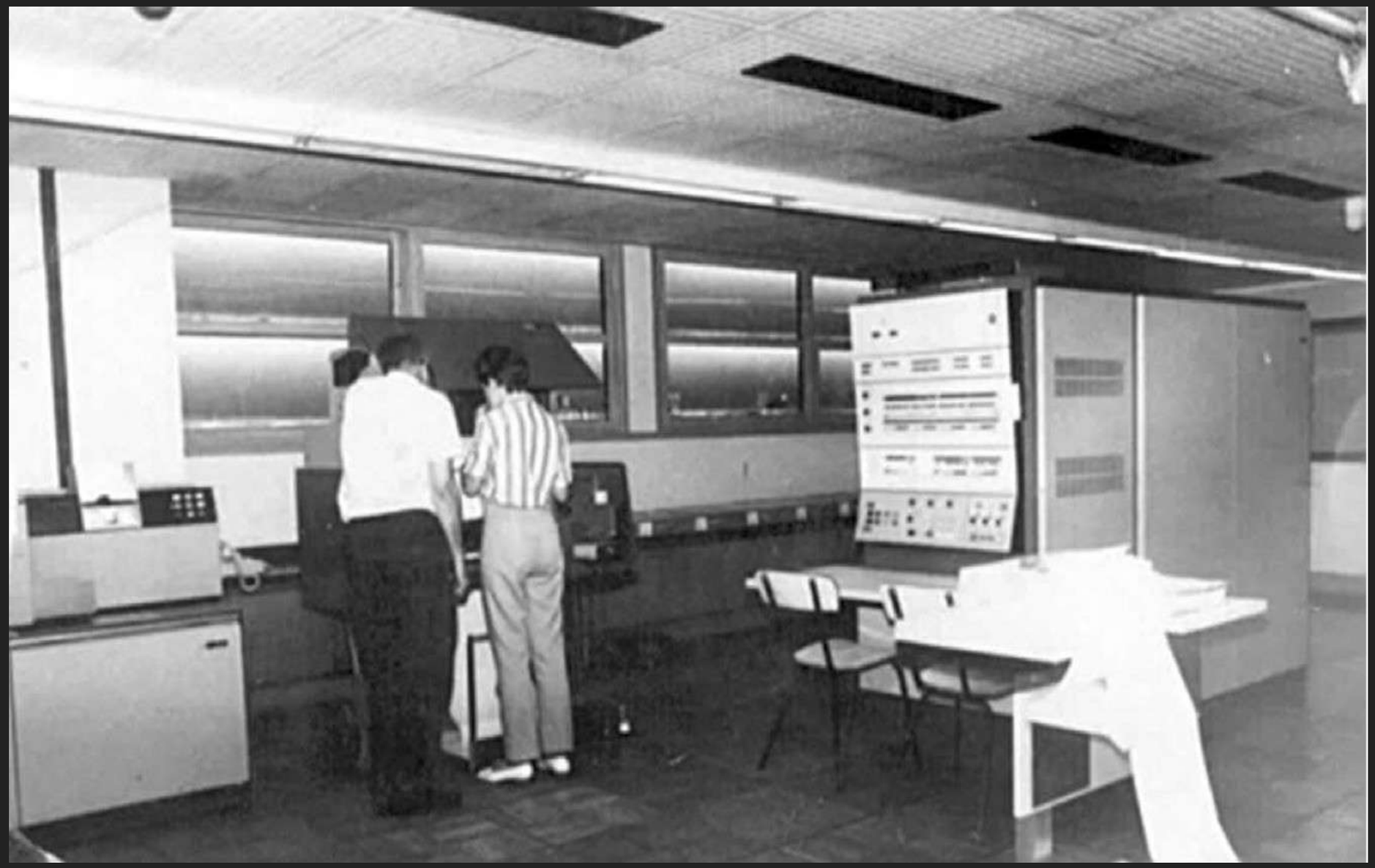
ÁLVARO VIDELA - @OLD_SOUND

LECTOR IN CÓDIGO

MICROSOFT

CLEMENTINA

CLEMENTINA



EXPLORE THE RELATION BETWEEN THE PROCESS OF WRITING COMPUTER PROGRAMS WITH THAT OF WRITING LITERARY WORKS OF FICTION.

UMBERTO ECO

LECTOR IN FABULUA

SIX WALKS IN THE FICTIONAL WOODS

WHAT CAN WE LEARN FROM THESE THEORIES TO BECOME BETTER* PROGRAMMERS

WHAT CAN WE LEARN FROM THESE THEORIES TO BECOME BETTER* PROGRAMMERS?

WHAT A PROGRAMMER DOES

It has been believed that a programmer occasionally writes code and gets it running on a computer, and that this is what he is paid for. In spite of his obvious inefficiency, no one else seems to do this work more effectively. However, his activity is still observed principally as loafing—a kind of ritual (like the British and teatime) which must be put up with.

Another view of what a programmer does addresses more constructively all that "wasted" time and cludes more than the running code, more than the symbolic code, or even the operator's guide, the maintenance guide, or the design guide. For in fact, in response to any serious breach of the program's integrity, a programmer will become involved, as part of the integral organization built by the original programmer. If one now looks closely, he can begin to recognize the intent of those steps in the ritual of programming.

WHAT A PROGRAMMER DOES

It has been believed that a programmer occasionally writes code and gets

cludes more than the running code, more than the symbolic code, or even

(like the British and teatime) which must be put up with.

Another view of what a programmer does addresses more constructively all that "wasted" time and

by the original programmer. If one now looks closely, he can begin to recognize the intent of those steps in the ritual of programming.

“A programmer does not primarily write code; rather, he primarily writes to another programmer about his problem solution”

“Programs must be written for people to read, and only incidentally for machineS to execute”

THE USE OF SUB-ROUTINES IN PROGRAMMES

D. J. Wheeler

Cambridge & Illinois Universities

THE USE OF SUB-ROUTINES IN PROGRAMMES

D. J. Wheeler

Cambridge & Illinois Universities

The above remarks may be summarized by saying sub-routines are very useful-although not absolutely necessary-and that the prime objectives to be born in mind when constructing them are simplicity of use, correctness of codes and accuracy of description. All complexities should-if possible -be buried out of sight.

LET'S TALK ABOUT NODEJS

```
10 PRINT CHR$(205.5+RND(1)); : GOTO 10
```

“The presence of these optional spaces indicates some concern for the people who will deal with this code, rather than merely the machine that will process it”

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`running at http://${hostname}:${port}/`);
});
```

```
const
http=require("http"),hostname="127.0.0.1",port=3e3,serv
er=http.createServer((e,t)=>{t.statusCode=200,t.setHead
er("Content-Type","text/plain"),t.end("Hello
World\n")});server.listen(3e3,hostname,
()=>{console.log("Server running at http://
127.0.0.1:3000/")});
```

LITERATURE AND PROGRAMMING

LITERATE PROGRAMMING

Donald Knuth

“Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do”

LITERATE PROGRAMMING

- Introduces the WEB system
- Write documentation along with the code
- Partially adopted by tools like JavaDocs and the like

EXPLAINS HOW WEB WORKS,
BUT NOT HOW TO WRITE CODE
THAT'S EASIER TO UNDERSTAND

CYBERTEXT: PERSPECTIVES ON ERGODIC LITERATURE

Aarseth, Espen J

"[. . .] a search for literary value in texts that are neither intended nor structured as literature will only obscure the unique aspects of these texts and transform a formal investigation into an apologetic crusade."

“Programs are normally written with two kinds of receivers in mind: the machines and other programmers. This gives rise to a double standard of aesthetics, often in conflict: efficiency and clarity”

"a difference between writing and programming, [is that] in programming, the programmer gets feedback very early on whether the program text is executable, during compiling. Furthermore, they get feedback on whether the program is working as intended"

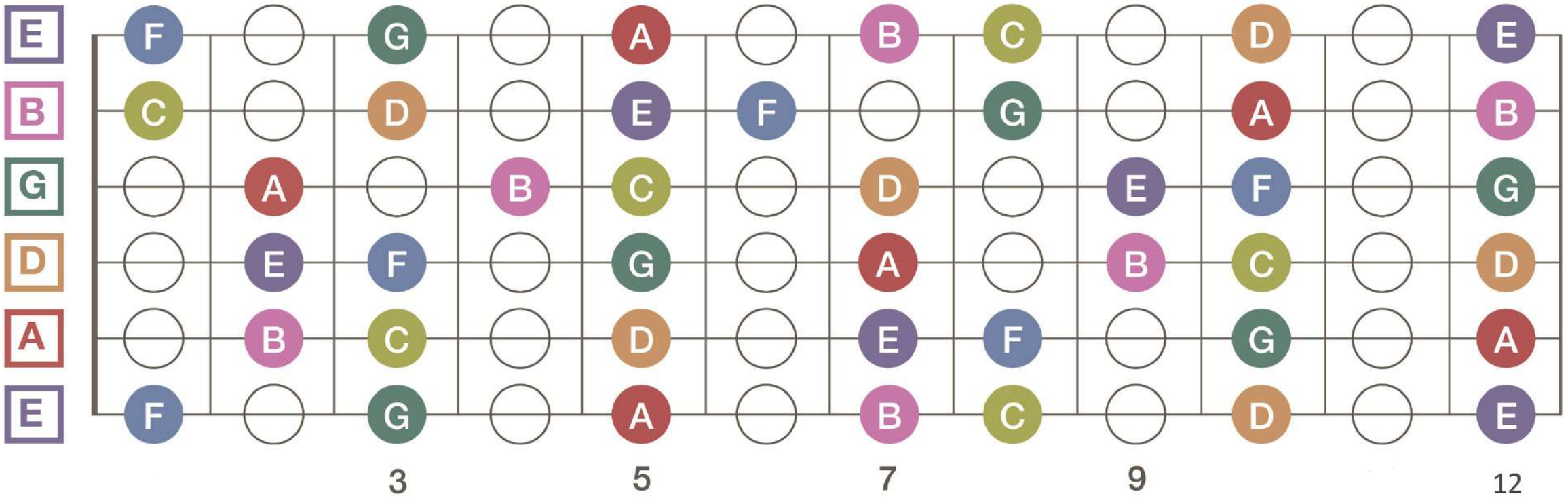
Hermans, Felienne, and Marlies Aldewereld

ABOUT EARLY FEEDBACK

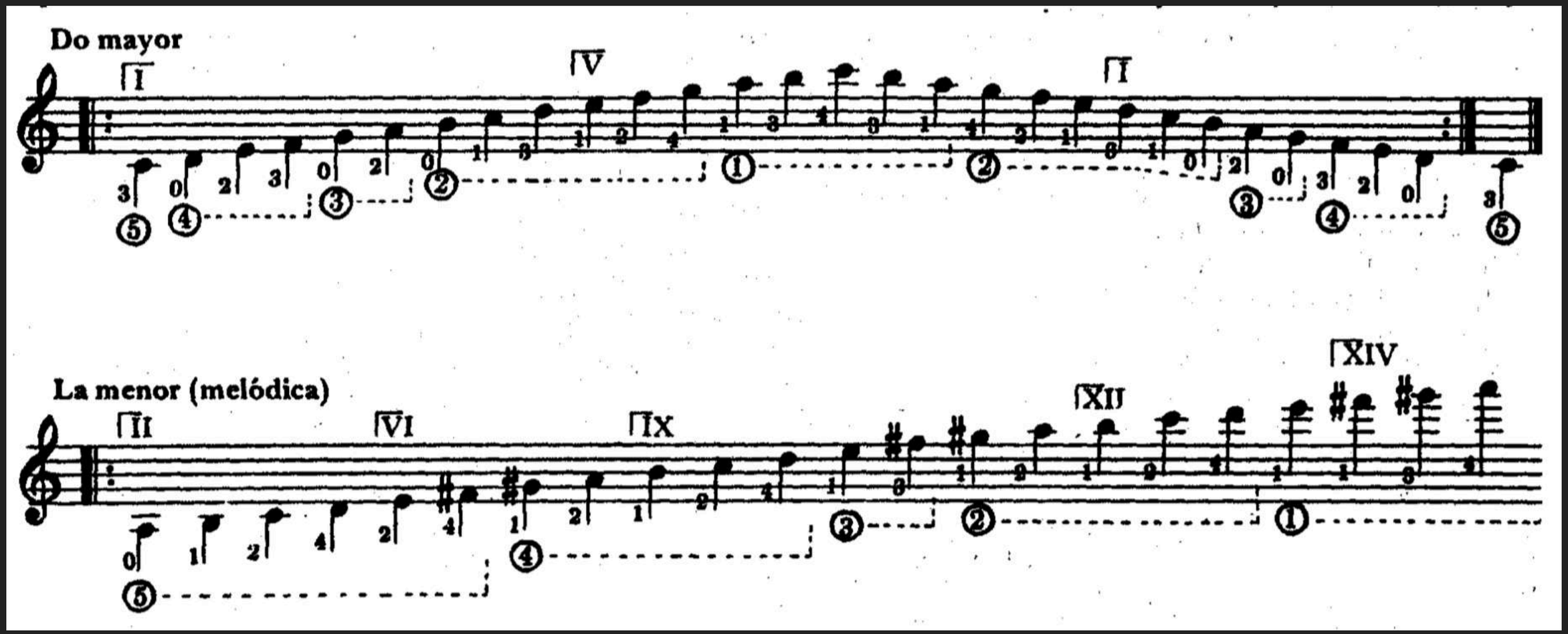
- What does the program means?
- What process of the real world is trying to represent?
- How the problem was solved?

COMPARE THIS WITH MUSIC INTERPRETATION

NOTES ON THE GUITAR



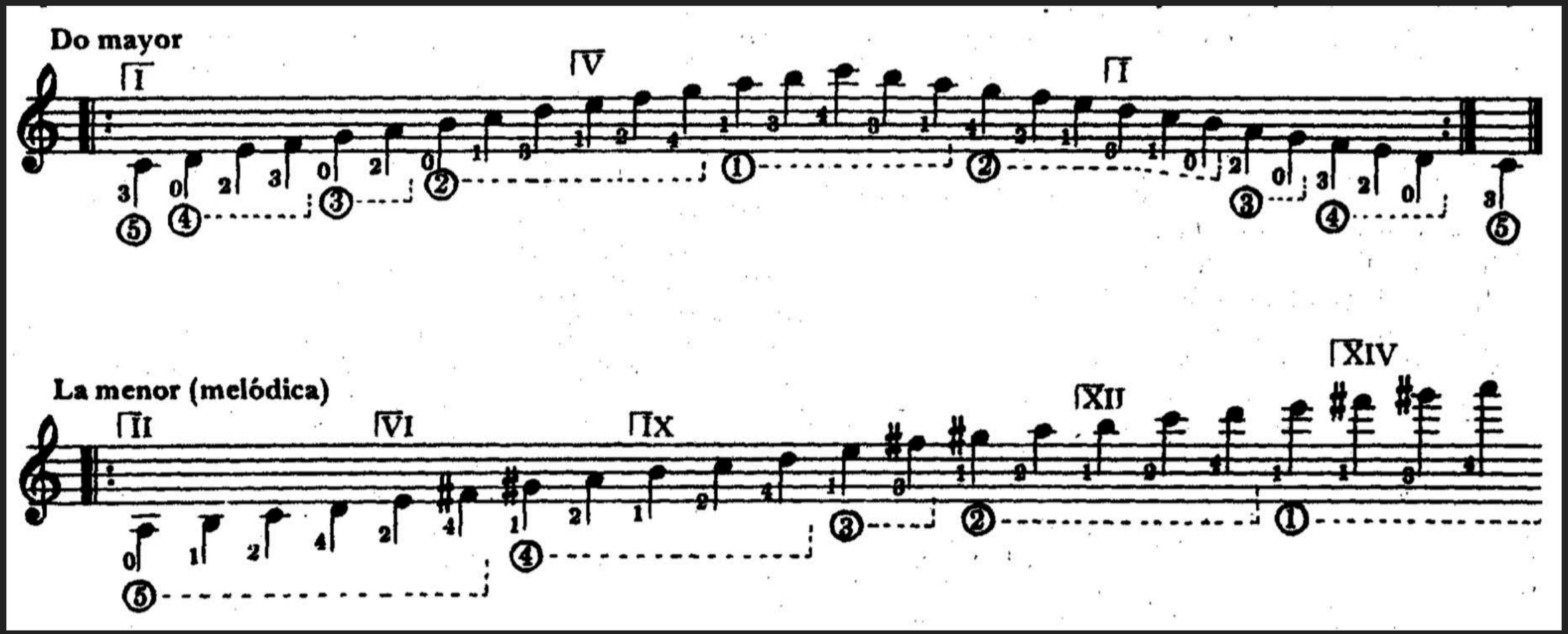
ABEL CARLEVARO



“CORRECT GUITAR PLAYING IS UNCONCEIVABLE WITHOUT CORRECT FINGERING”

Abel Carlevaro

ABEL CARLEVARO



ABOUT EARLY FEEDBACK

- Knuth: Is 2 a random number?
- Is a `square` function that returns a hardcoded `25` a correct implementation?
- As long as we provide `[5, -5]` as arguments, it is correct.
- TDD advocates this kind of program building

“Program testing can be used to show the presence of bugs, but never to show their absence!”

Edsger Dijkstra

ABOUT EARLY FEEDBACK

- Knuth: Is 2 a random number?
- Is a `square` function that returns a hardcoded `25` a correct implementation?
- As long as we provide `[5, -5]` as arguments, it is correct
- TDD advocates this kind of program building
- QuickCheck tries to alleviate this problem

HOW CAN WE SHARE KNOWLEDGE BETWEEN PROGRAMMERS?

“THE CODE SPEAKS FOR ITSELF”

WE ARE NOT ADVERSARIES

IMAGINE IF EVERY TIME WE TRIED TO READ A BOOK, WE HAD TO PLAY CODE BREAKERS?

UNLESS WE WERE READING FINNEGANS WAKE...

PROGRAMMING AS THEORY BUILDING

Peter Naur

"[. . .] A PERSON WHO HAS OR POSSESSES A THEORY IN THIS SENSE KNOWS HOW TO DO CERTAIN THINGS AND IN ADDITION CAN SUPPORT THE ACTUAL DOING WITH EXPLANATIONS, JUSTIFICATIONS, AND ANSWERS TO QUERIES, ABOUT THE ACTIVITY OF CONCERN"

“[. . .] WHAT HAS TO BE BUILT BY THE PROGRAMMER IS A THEORY OF HOW CERTAIN AFFAIRS OF THE WORLD WILL BE HANDLED BY, OR SUPPORTED BY, A COMPUTER PROGRAM”

THIS THEORY IS VERY HARD TO SHARE. IT WON'T BE REFLECTED IN DOCUMENTATION OR THE PROGRAM TEXT

HOW CAN WE SHARE THIS THEORY?

THE ENCYCLOPEDIA

THE ENCYCLOPEDIA

- There’s the Encyclopedia
- And there’s the “encyclopedia”
- All the world’s knowledge vs. my knowledge

“THE COMPETENCE OF THE DESTINATARY IS NOT NECESSARILY THAT OF THE SENDER”

ABSENCE OF DETAILS

WE FILL IN DETAILS FROM OUR OWN ENCYCLOPEDIA

PARASITICAL WORLDS

“fictional worlds are parasitical worlds because, if alternative properties are not spelled out, we take for granted the properties holding in the real world”

THE MODEL READER

MODEL READER

- Not the empirical reader
- Lives in the mind of the author (the empirical one)
- It’s built as the author writes the story
- Helps the author decide how much detail to include in the story

TEXTUAL COOPERATION

NOTICE

Dogs must be carried on escalator

DOGS MUST BE CARRIED ON ESCALATOR

- Does it mean that you must carry a dog in the escalator?
- Are you going to be banned from the escalator unless you find a stray dog to carry?
- “Carried” is to be taken metaphorically and help dogs get through life?

DOGS MUST BE CARRIED ON ESCALATOR

- How do I know this is not a decoration?
- I need to understand that the sign has been placed there by some authority
- Conventions: I understand that “escalator” means *this* escalator and not some escalator in Paraguay
- “Must be” means must be *now*

“A text is a lazy (or economic) mechanism that lives on the surplus value of meaning introduced by the recipient”

“A TEXT WANTS SOMEONE TO HELP IT WORK”

READING IS ESSENTIALLY A WORK OF COOPERATION BETWEEN THE AUTHOR AND THE READER

A STRATEGIC GAME BETWEEN AUTHOR AND READER

NAPOLEON VS WELLINGTON

BOURDIEU & TEXTUAL DEVICES

DEVICES TO HELP PROGRAMMERS

- Type declarations
- Documentation
- Paratexts

PARATEXTS

"THE "PARATEXT" CONSISTS OF THE WHOLE SERIES OF MESSAGES THAT ACCOMPANY AND HELP EXPLAIN A GIVEN TEXT—MESSAGES SUCH AS ADVERTISEMENTS, JACKET COPY, TITLE, SUBTITLES, INTRODUCTION, REVIEWS, AND SO ON."

Eco quoting Genette

“TO INDICATE WHAT IS AT STAKE, WE CAN ASK ONE SIMPLE QUESTION AS AN EXAMPLE: LIMITED TO THE TEXT ALONE AND WITHOUT A GUIDING SET OF DIRECTIONS, HOW WOULD WE READ JOYCE'S ULYSSES IF IT WERE NOT ENTITLED ULYSSES?”

Gérard Genette

PARATEXTS IN CODE

- Documentation
- package names
- folder structure
- pragmas (as in Haskell)
- imports (hiding things from the Prelude or overloading it)
- compiler flags
- running mode (test, production, benchmarks)

A PRIVILEGED PLACE OF A PRAGMATICS AND A STRATEGY, OF AN INFLUENCE ON THE PUBLIC, AN INFLUENCE THAT – WHETHER WELL OR POORLY UNDERSTOOD AND ACHIEVED – IS AT THE SERVICE OF A BETTER RECEPTION FOR THE TEXT AND A MORE PERTINENT READING OF IT

Gérard Genette

KEEPING PARATEXTS RELEVANT

HOW TO KEEP COMMENTS UP TO DATE?

NOT EVEN CERVANTES
ESCAPED THIS FATE

IN DON QUIXOTE, THE ORIGINAL DESCRIPTION FOR CHAPTER X DOESN'T MATCH THE CONTENTS OF THE CHAPTER!

CONSIDER THIS CODE

```
class User {
    String username;
    String password;
    String role;

    User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role     = role;
    }

    public String getUsername() {return username;}
    public String getPassword() {return password;}
    public String getRole()     {return role;}
}
```

```
User user = new User('alice', 'secret', 'admin');
assertEquals(user.getUsername(), 'alice');
assertEquals(user.getPassword(), 'secret');
assertEquals(user.getRole(), 'admin');
```

THE PREVIOUS TEST CAN GIVE US FEEDBACK ABOUT THE CODE WORKING AS EXPECTED, BUT WE ARE STILL IN THE DARK ABOUT WHAT IS THIS CLASS PURPOSE, THAT IS, WHAT CONCEPT OF THE REAL WORLD THIS CLASS IS TRYING TO REPRESENT.

```
class User {
    String username;
    String password;
    String role;

    User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role     = role;
    }

    public String getUsername() {return username;}
    public String getPassword() {return password;}
    public String getRole()     {return role;}
}
```

```
package database;

class User {
    String username;
    String password;
    String role;

    User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role     = role;
    }

    public String getUsername() {return username;}
    public String getPassword() {return password;}
    public String getRole()     {return role;}
}
```

```
package model;

class User {
    String username;
    String password;
    String role;

    User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role     = role;
    }

    public String getUsername() {return username;}
    public String getPassword() {return password;}
    public String getRole()     {return role;}
}
```

```
class Person {
    String name;
    String age;

    User(String name, String age) {
        this.name = name;
        this.age  = age;
    }

    public String getName() {return name;}
    public String getAge()  {return age;}
}
```

```
// This is not a person
class Person {
    String name;
    String age;

    User(String name, String age) {
        this.name = name;
        this.age  = age;
    }

    public String getName() {return name;}
    public String getAge()  {return age;}
}
```

HOW TO BUILD THE MODEL READER FOR OUR CODE?

METAPHORS

CHOOSING THE RIGHT DATA STRUCTURE

CHOOSE THE RIGHT DATA STRUCTURE

CHOOSE THE RIGHT DATA STRUCTURE

- Array

CHOOSE THE RIGHT DATA STRUCTURE

- Array
- Set

CHOOSE THE RIGHT DATA STRUCTURE

- Array
- Set
- LinkedList

CHOOSE THE RIGHT DATA STRUCTURE

- Array
- Set
- LinkedList
- Queue

CHOOSE THE RIGHT DATA STRUCTURE

- Array
- Set
- LinkedList
- Queue
- Stack

A PROGRAM'S EXPLANATORY POWER IS THE MEASURE OF ITS OWN ELEGANCE

DATA STRUCTURES HAVE EXPLANATORY POWER

COGNITIVE LEAPS

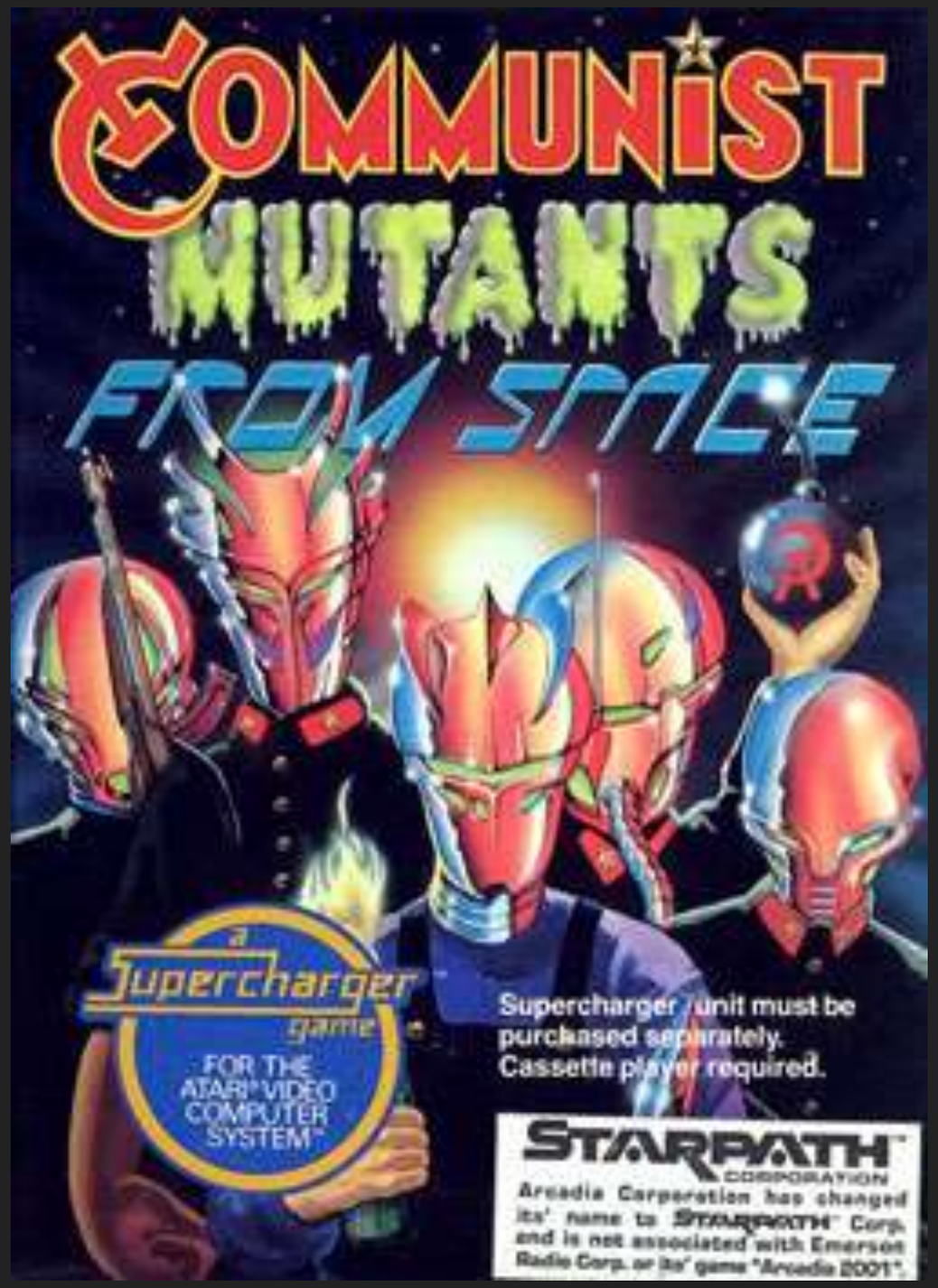
CLEAN CODE

CLEAN CODE?

CLEAN CODE DOESN'T EXIST

CIAO

"Hegemonic culture propagates its own values and norms so that they become the "common sense" values of all and thus maintain the status quo"



ARE YOU A COMMU NIST!?

ARE YOU AGAINST BELKA & STRELKA?

CLEAN CODE

CLEAN CODE

- Requires a shared encyclopedia
- Shared reading competencies
- Old by definition

MODES OF INTERPRETATION

“Semantic interpretation is the result of the process by which the reader, facing a Linear Text Manifestation, fills it up with a given meaning.”

“Critical Interpretation is, on the contrary, a metalinguistic activity which aims at describing and explaining for which formal reasons a given text produces a given response.”

LET'S GET CRITICAL

THANK YOU

@old_sound

Lector in Codigo or the role of the reader

Full Text: PDF Get this Article

Author: Alvaro Videla Durazno, Uruguay

2018 Article

Published in:

· Proceeding

Programming'18 Companion Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming

Pages 180-186

Nice, France — April 09 - 12, 2018

ACM New York, NY, USA ©2018

table of contents ISBN: 978-1-4503-5513-1

doi>10.1145/3191697.3214326

Bibliometrics

· Citation Count: 2
· Downloads (cumulative): 23
· Downloads (12 Months): 14
· Downloads (6 Weeks): 2

Metaphors We Compute By

Code is a story that explains how to solve a particular problem

Alvaro Videla

In their now-classic book *Metaphors We Live By*,[6] George Lakoff and Mark Johnson set out to show the linguistic and philosophical worlds that metaphor isn't just a matter of poetry and rhetorical flourish. They presented how metaphor permeates all areas of our lives, and in particular that metaphor dictates how we understand the world, how we act in it, how we live in it. They showed that our conceptual system is based on metaphors, too, but since we are not normally aware of our own conceptual system, they had to study it via a proxy: language.

In the Beginning was the Word

By studying language, Lakoff and Johnson tried to understand how metaphors work by imposing meaning in our lives. The basic example they present is the conceptual metaphor "argument is war." We understand the act of arguing with another person in the same way we understand war. This leads to the following expressions in our daily language:

- Your claims are *indefensible*.
- He *attacked every weak point* in my argument.
- I *demolished* his argument.
- I never *won* an argument with him.

These sentences may seem innocuous, but the problem is how we act and feel based on them. We end up seeing the person we are arguing with as our

ALVARO VIDELA

Programming as translation

Converting the real world into digital abstractions requires distillation. And, like literary translators, developers must understand their biases.

PART OF
ISSUE 8
FEB 2019
Internationalization

> What does it mean 'to translate'? A quick answer could be:
> to say the same thing in a different language.
> — Umberto Eco

Metaphor is a powerful tool for approaching new problems and finding creative solutions to them. Let's use a framing metaphor: What could we learn from looking at programming as translation?

More specifically, programming translates domain problems — a hardware store inventory, a public library catalogue, a ticket reservation system — into computer programs.

Many factors come into play when we adapt a system from the real world into the digital world. Converting the analog into the digital requires discretization, leaving things out. What we filter out — or what we focus on — depends on our biases. How do conventional translators handle issues of bias? What can programmers learn from them?

ALVARO VIDELA

Notes on the synthesis of labyrinths

Solving problems in software development is not unlike finding your way out of a maze. Consider how documentation might reflect the twists and turns you faced along way—not just the end result.

PART OF
ISSUE 6
AUG 2018
Documentation

> *No one realized that the book and the labyrinth were one and the same.*
> — Jorge Luis Borges

One morning you arrive at the office to find your manager waiting for you with a new task: She wants you to choose a JavaScript framework for the company. All new projects will be built using the library of your choice. What a responsibility!

A quick internet search reveals a plethora of frameworks to choose from. You land on a website that compares their pros and cons. From there, you decide to further explore the two most popular ones: Let's call them reaction.js and view.js.

As soon as you dive into reaction.js, you notice that you need to learn its XML markup language. Soon, your browser has five open tabs just for this framework, and you're

REFERENCES

- Aarseth, Espen J. Cybertext: Perspectives on Ergodic Literature. Johns Hopkins University Press, 1997.
- Beck, Kent. Test-Driven Development: by Example. Addison-Wesley, 2006.
- Berger, Peter L., and Thomas Luckmann. The Social Construction of Reality: a Treatise in the Sociology of Knowledge. Penguin, 1991.
- Borges, Jorge Luis, and Andrew Hurley. Collected Fictions. Penguin Books, 1999.

REFERENCES

- Carlevaro, Abel. Serie Didactica: Para Guitarra. Barry, 1966.
- Eagleton, Terry. Literary Theory: an Introduction. Blackwell Publishing, 2015.
- Eco, Umberto, and Anthony Oldcorn. From the Tree to the Labyrinth: Historical Studies on the Sign and Interpretation. Harvard University Press, 2014.
- Eco, Umberto. Lector in Fabula: La Cooperazione Interpretativa Nei Testi Narrativi. Bompiani, 2016.

REFERENCES

- Eco, Umberto. Six Walks in the Fictional Woods. Harvard Univ. Press, 2004.
- Genette, Gérard. Paratexts: Thresholds of Interpretation. Cambridge Univ. Press, 2001.
- Gärdenfors, Peter. Geometry of Meaning: Semantics Based on Conceptual Spaces. The MIT Press, 2017.
- Hermans, Felienne, and Marlies Aldewereld. “Programming Is Writing Is Programming.” Proceedings of the International Conference on the Art, Science, and Engineering of Programming - Programming '17, 2017, doi:10.1145/3079368.3079413.

REFERENCES

- Kent, William, and Steve Hoberman. Data and Reality: a Timeless Perspective on Perceiving and Managing Information in Our Imprecise World. Technics Publications, 2012.
- Lewis, James, and Martin Fowler. “Microservices.” Martinfowler.com, 25 Mar. 2014, martinfowler.com/articles/microservices.html.
- Moore. “What a Programmer Does.” Datamation, Apr. 1967, pp. 177–178., archive.computerhistory.org/resources/text/Knuth_Don_X4100/PDF_index/k-9-pdf/k-9-u2769-1-Baker-What-Programmer-Does.pdf.
- Naur, Peter. “Programming as Theory Building.” Microprocessing and Microprogramming, vol. 15, no. 5, 1985, pp. 253–261., doi:10.1016/0165-6074(85)90032-8.

REFERENCES

- “Random Numbers.” The Art of Computer Programming, by Donald Ervin Knuth, vol. 2, Addison-Wesley, 2011.
- Steele, Julie, and Noah P. N. Iliinsky. Beautiful Visualization. O'Reilly, 2010.
- Videla, Alvaro. “Metaphors We Compute By.” Communications of the ACM, vol. 60, no. 10, 2017, pp. 42–45., doi:10.1145/3106625.