Ingeniería de Software I
Diseño de Software con Objetos
Parte I
UBA - FCEyN

Hernán Wilkinson

Twitter: @HernanWilkinson

Un poco de historia…

Programación Estructurada, ¿cuándo?

A.P.I.C. Studies in Data Processing
No. 8

STRUCTURED PROGRAMMING

O.-J. DAHL
Universitet i Oslo,
Matematisk Institut,
Blindern, Oslo, Norway

E. W. DIJKSTRA
Department of Mathematics,
Technological University,
Eindhoven, The Netherlands

C. A. R. HOARE
Department of Computer Science,
The Queen's University of Belfast,
Belfast, Northern Ireland

A.P.I.C. Studies in Data Processing
No. 8

STRUCTURED PROGRAMMING

O.-J. DAHL
Universitet i Oslo,
Matematisk Institut,
Blindern, Oslo, Norway

E. W. DIJKSTRA
Department of Mathematics,
Technological University,
Eindhoven, The Netherlands

C. A. R. HOARE
Department of Computer Science,
The Queen's University of Belfast,
Belfast, Northern Ireland

ACADEMIC PRESS INC. (LONDON) LTD.
24/28 Oval Road,
London NW1

United States Edition published by
ACADEMIC PRESS INC.
111 Fifth Avenue
New York, New York 10003

Second printing 1973

Programación Orientada a Objetos, ¿cuándo?

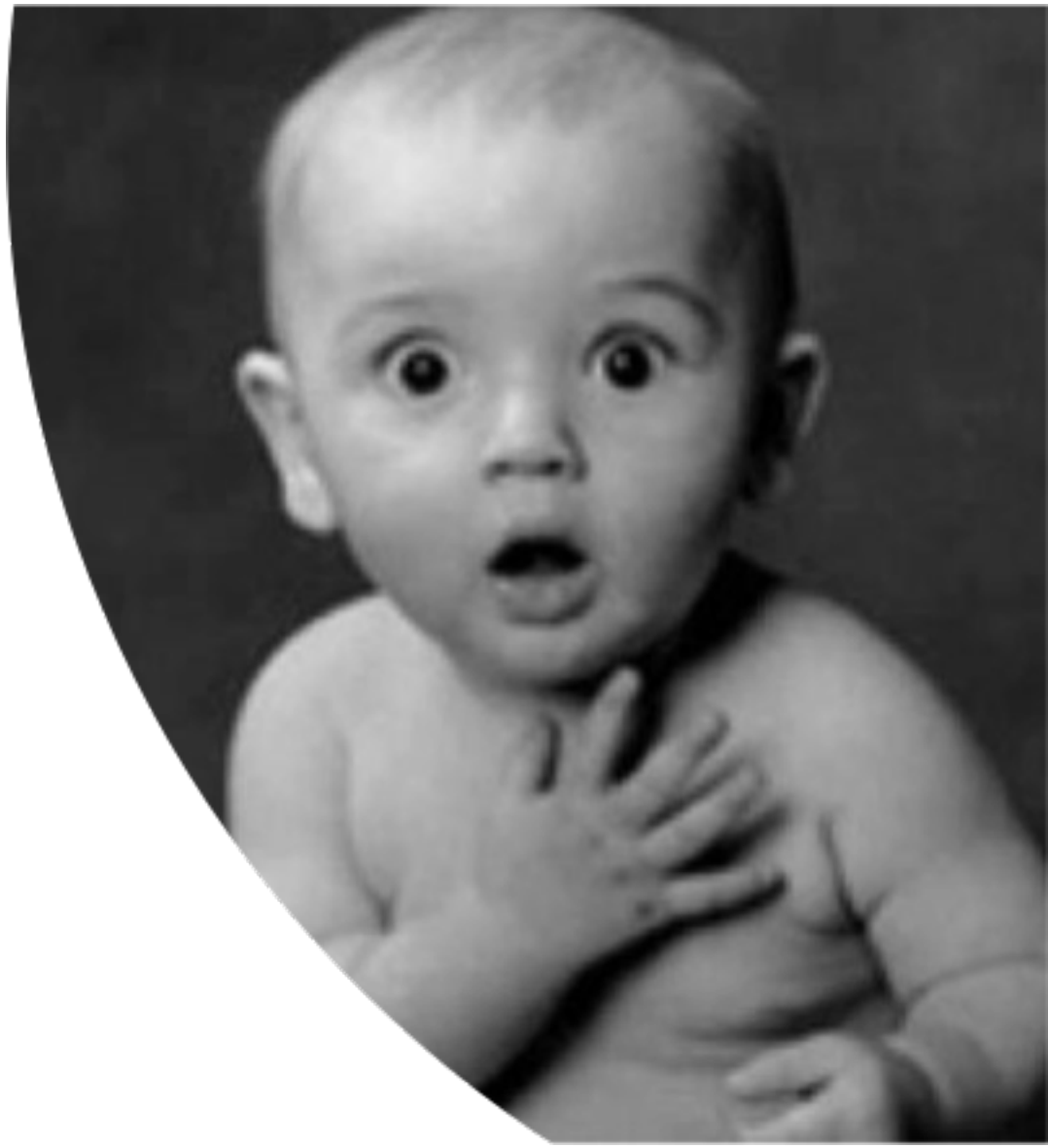
Simula 67

Ole-Johan Dahl y Kristen Nygaard

Simula67

Ole-Johan Dahl y Kristen Nygaard

OOP antes que Structured Programming??



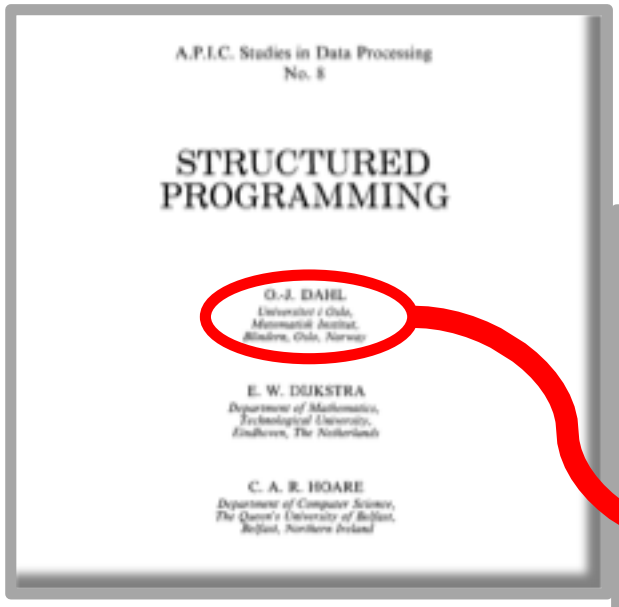
A.P.I.C. Studies in Data Processing
No. 8

STRUCTURED PROGRAMMING

O.-J. DAHL
Universitet i Oslo,
Matematisk Institut,
Blindern, Oslo, Norway

E. W. DIJKSTRA
Department of Mathematics,
Technological University,
Eindhoven, The Netherlands

C. A. R. HOARE
Department of Computer Science,
The Queen's University of Belfast,
Belfast, Northern Ireland



A.P.I.C. Studies in Data Processing
No. 8

STRUCTURED PROGRAMMING

O.-J. DAHL
Universitet i Oslo, Matematisk Institut, Blindern, Oslo, Norway

E. W. DIJKSTRA
Department of Mathematics, Technological University, Eindhoven, The Netherlands

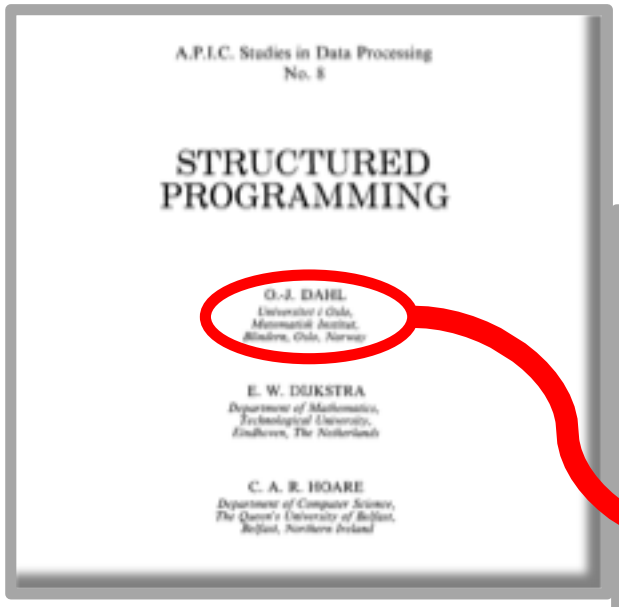
C. A. R. HOARE
Department of Computer Science, The Queen's University of Belfast, Belfast, Northern Ireland

Page

12. Axiomatisation 166
References 174

III. **Hierarchical Program Structures.** OLE-JOHAN DAHL AND C. A. R. HOARE 175

1. Introduction 175
2. Preliminaries 175
3. Object Classes 179
4. Coroutines 184
5. List Structures 193
6. Program Concatenation 202
7. Concept Hierarchies 208
References 220



A.P.I.C. Studies in Data Processing
No. 8

STRUCTURED PROGRAMMING

O.-J. DAHL
Universitet i Oslo, Matematisk Institut, Blindern, Oslo, Norway

E. W. DIJKSTRA
Department of Mathematics, Technological University, Eindhoven, The Netherlands

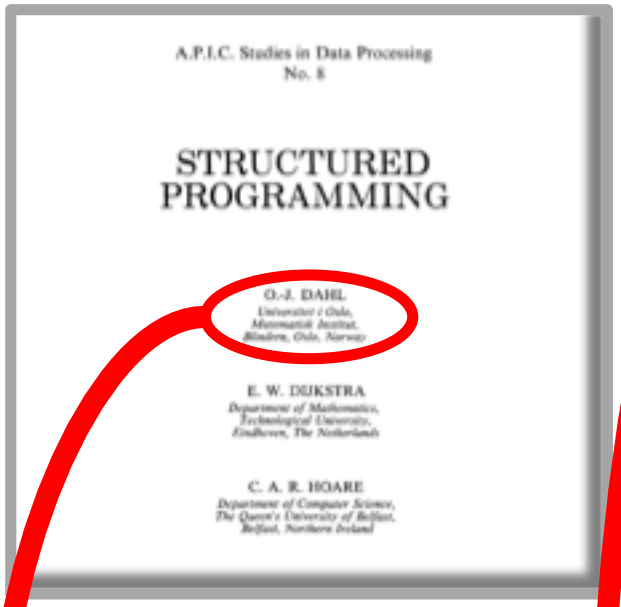
C. A. R. HOARE
Department of Computer Science, The Queen's University of Belfast, Belfast, Northern Ireland

Page

12. Axiomatisation 166
References 174

III. Hierarchical Program Structures. OLE-JOHAN DAHL AND C. A. R. HOARE 175

1. Introduction 175
2. Preliminaries 175
3. Object Classes 179
4. Coroutines 184
5. List Structures 193
6. Program Concatenation 202
7. Concept Hierarchies 208
References 220



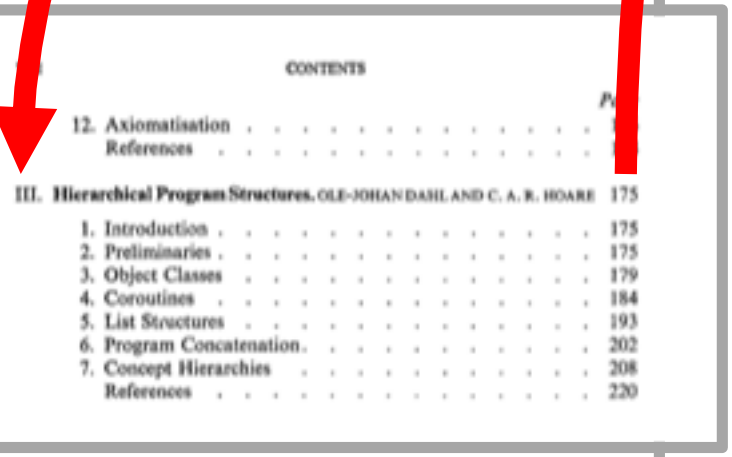
A.P.I.C. Studies in Data Processing
No. 8

STRUCTURED PROGRAMMING

O.-J. DAHL
Universitet i Oslo,
Matematisk Institut,
Blindern, Oslo, Norway

E. W. DIJKSTRA
Department of Mathematics,
Technological University,
Eindhoven, The Netherlands

C. A. R. HOARE
Department of Computer Science,
The Queen's University of Belfast,
Belfast, Northern Ireland



CONTENTS

12. Axiomatisation 175
References 175

III. **Hierarchical Program Structures.** OLE-JOHAN DAHL AND C. A. R. HOARE 175

1. Introduction 175
2. Preliminaries 175
3. Object Classes 179
4. Coroutines 184
5. List Structures 193
6. Program Concatenation 202
7. Concept Hierarchies 208
References 220

III. Hierarchical Program Structures

OLE-JOHAN DAHL AND C. A. R. HOARE

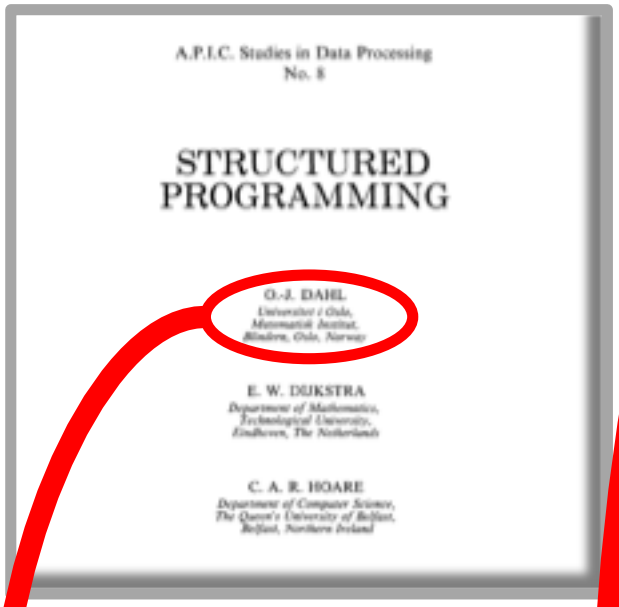
1. INTRODUCTION

In this monograph we shall explore certain ways of program structuring and point out their relationship to concept modelling.

We shall make use of the programming language SIMULA 67 with particular emphasis on structuring mechanisms. SIMULA 67 is based on ALGOL 60 and contains a slightly restricted and modified version of ALGOL 60 as a subset. Additional language features are motivated and explained informally when introduced. The student should have a good knowledge of ALGOL 60 and preferably be acquainted with list processing techniques.

For a full exposition of the SIMULA language we refer to the "Simula 67 Common Base Language" [2]. Some of the linguistic mechanisms introduced in the monograph are currently outside the "Common Base"*.

The monograph is an extension and reworking of a series of lectures given by Dahl at the NATO Summer School on Programming, Marktoberdorf 1970. Some of the added material is based on programming examples that have occurred elsewhere [3, 4, 5].



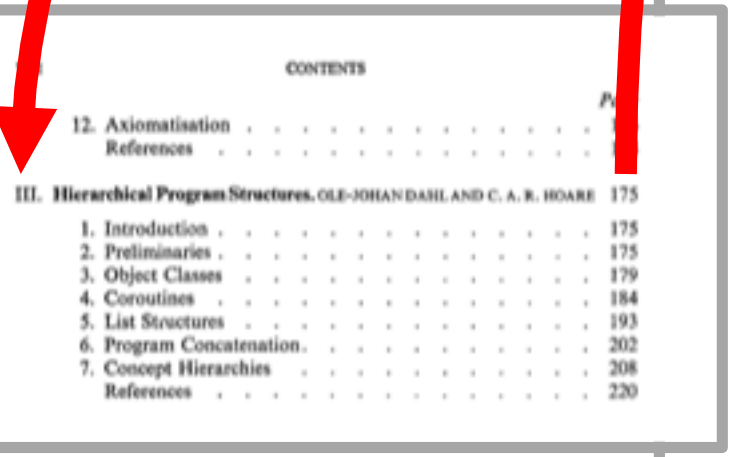
A.P.I.C. Studies in Data Processing
No. 8

STRUCTURED PROGRAMMING

O.-J. DAHL
Universitet i Oslo,
Matematisk Institut,
Blindern, Oslo, Norway

E. W. DIJKSTRA
Department of Mathematics,
Technological University,
Eindhoven, The Netherlands

C. A. R. HOARE
Department of Computer Science,
The Queen's University of Belfast,
Belfast, Northern Ireland



CONTENTS

12. Axiomatisation . . . . . [illegible]
References . . . . . [illegible]

III. **Hierarchical Program Structures.** OLE-JOHAN DAHL AND C. A. R. HOARE 175

1. Introduction 175
2. Preliminaries 175
3. Object Classes 179
4. Coroutines 184
5. List Structures 193
6. Program Concatenation 202
7. Concept Hierarchies 208
References 220

III. Hierarchical Program Structures

OLE-JOHAN DAHL AND C. A. R. HOARE

1. INTRODUCTION

In this monograph we shall explore certain ways of program structuring and point out their relationship to concept modelling.

We shall make use of the programming language SIMULA 67 with particular emphasis on structuring mechanisms. SIMULA 67 is based on ALGOL 60 and contains a slightly restricted and modified version of ALGOL 60 as a subset. Additional language features are motivated and explained informally when introduced. The student should have a good knowledge of ALGOL 60 and preferably be acquainted with list processing techniques.

For a full exposition of the SIMULA language we refer to the "Simula 67 Common Base Language" [2]. Some of the linguistic mechanisms introduced in the monograph are currently outside the "Common Base"*.

The monograph is an extension and reworking of a series of lectures given by Dahl at the NATO Summer School on Programming, Marktoberdorf 1970. Some of the added material is based on programming examples that have occurred elsewhere [3, 4, 5].

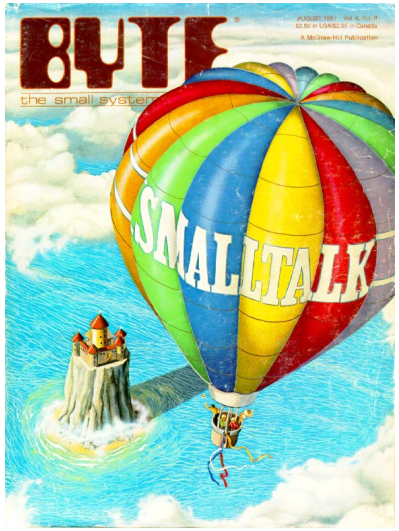
Paradigma de Objetos: Definiciones

- “Fundacional” ← Vamos a estudiar este!
  - También llamada Pura (no me gusta)
  - Más orientada hacia Lambda Calculus
  - Más orientada a idea de “modelo”
  - Smalltalk, Self, Ruby se acerca…
- No Fundacional
  - Más orientada a “máquina”
  - Más orientada a Turing
  - C++, Java, C#, etc.

Paradigma de Objetos

- Historia
  - Simula 67 (Nygrard y Dahl)
    - Antes que paradigma Estrucuturado
    - Goto Considered Harmfull – 68
    - Structured Programming – 71
    - (using Simula 67 as prog. lang.!!)
  - Smalltalk
    - Alan Kay
    - Dan Ingalls
    - Adele Goldberg

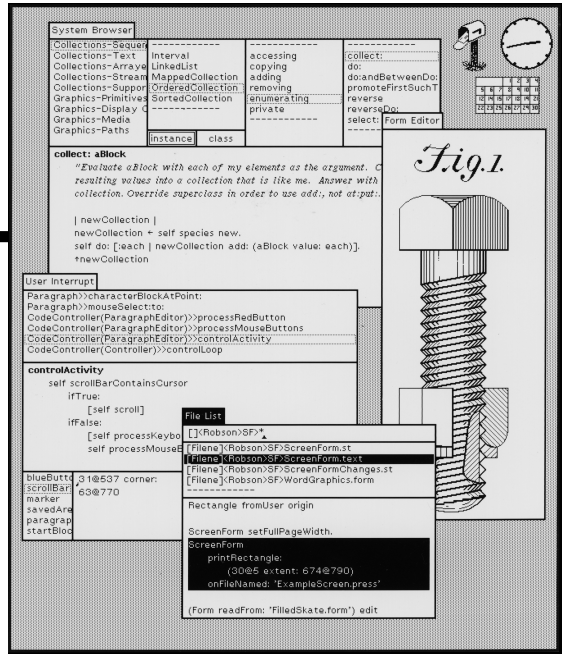
Dahl and Nygaard at the time of Simula's development



Paradigma de Objetos

DynaBook

Augment Children Comprehention



GUI - IDE

Smalltalk
(72,74,76,78,80)

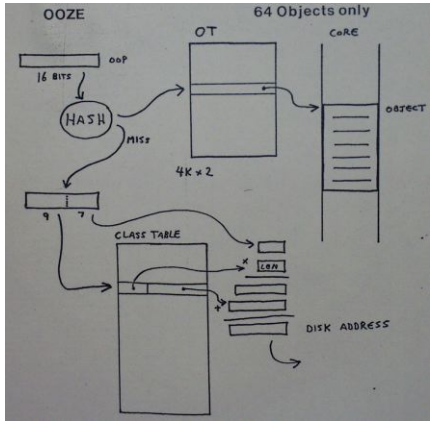
Object Oriented

Simula 67

Lisp

Flex Machine

VM



http://www.youtube.com/watch?v=AuXCc7WSczM

Paradigma de Objetos

- Alan Kay
  - "The best way to predict the future is to invent it"
  - "I invented the term Object-Oriented and I can tell you I did not have C++ in mind.“
  - "Java and C++ make you think that the new ideas are like the old ones. Java is the most distressing thing to hit computing since MS-DOS.“

http://video.google.com/videoplay?docid=-2950949730059754521

Paradigma de Objetos

- Principios Originales:
  - Simplicidad
  - Consistencia
  - Concretitud
  - Inmediate Feedback
  - Hacer Frente a la Complejidad

http://vimeo.com/36579366

Paradigma de Objetos

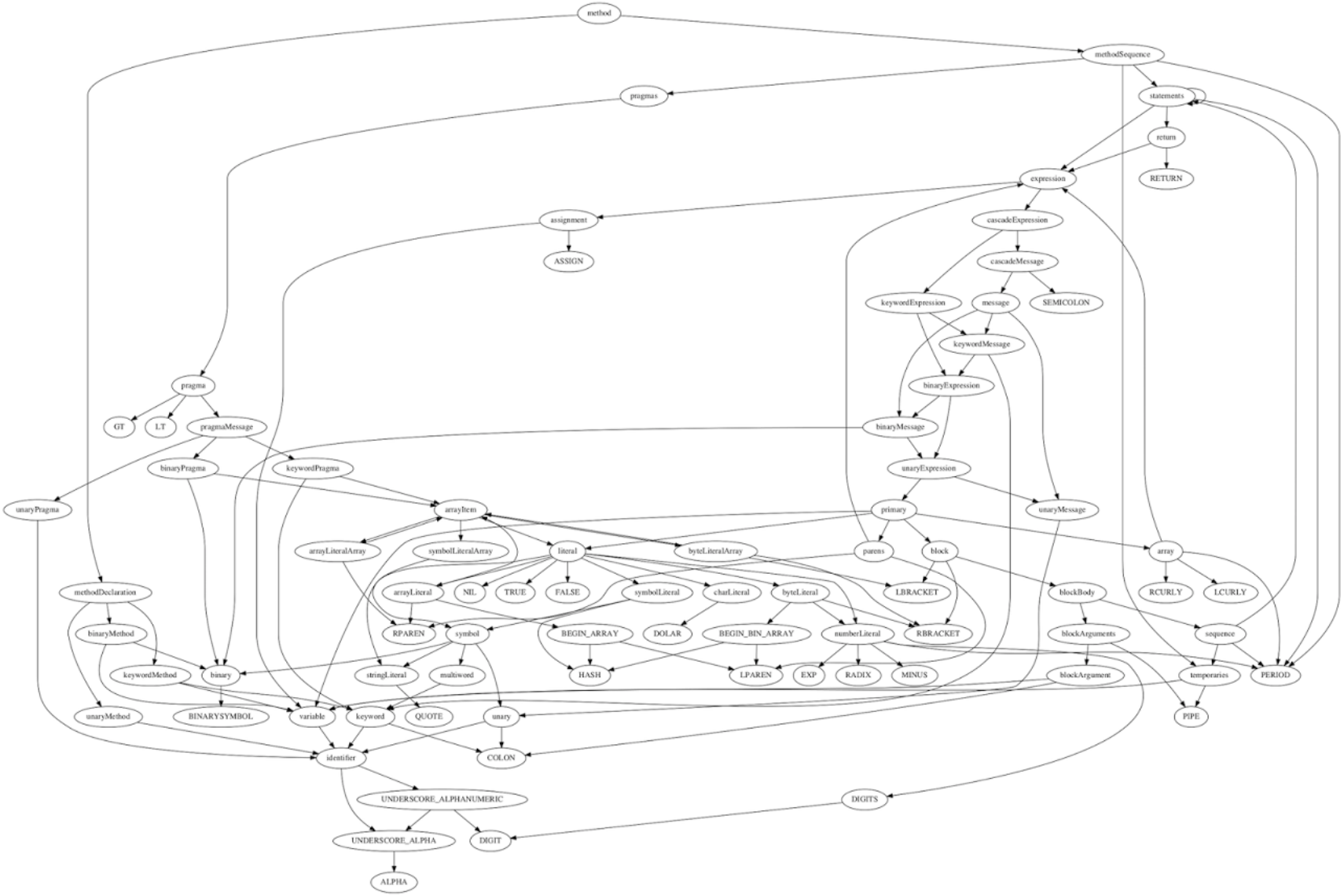
Complejidad = Esencial + Accidental

- Esencial:
  - Tiene que ver con el problema en si
  - No se puede reducir más allá de la complejidad del problema
- Accidental:
  - Tiene que ver con las herramientas que uso, en nuestro caso Lenguaje de Programación
  - Tengo que minimizarlo para minimizar Complejidad

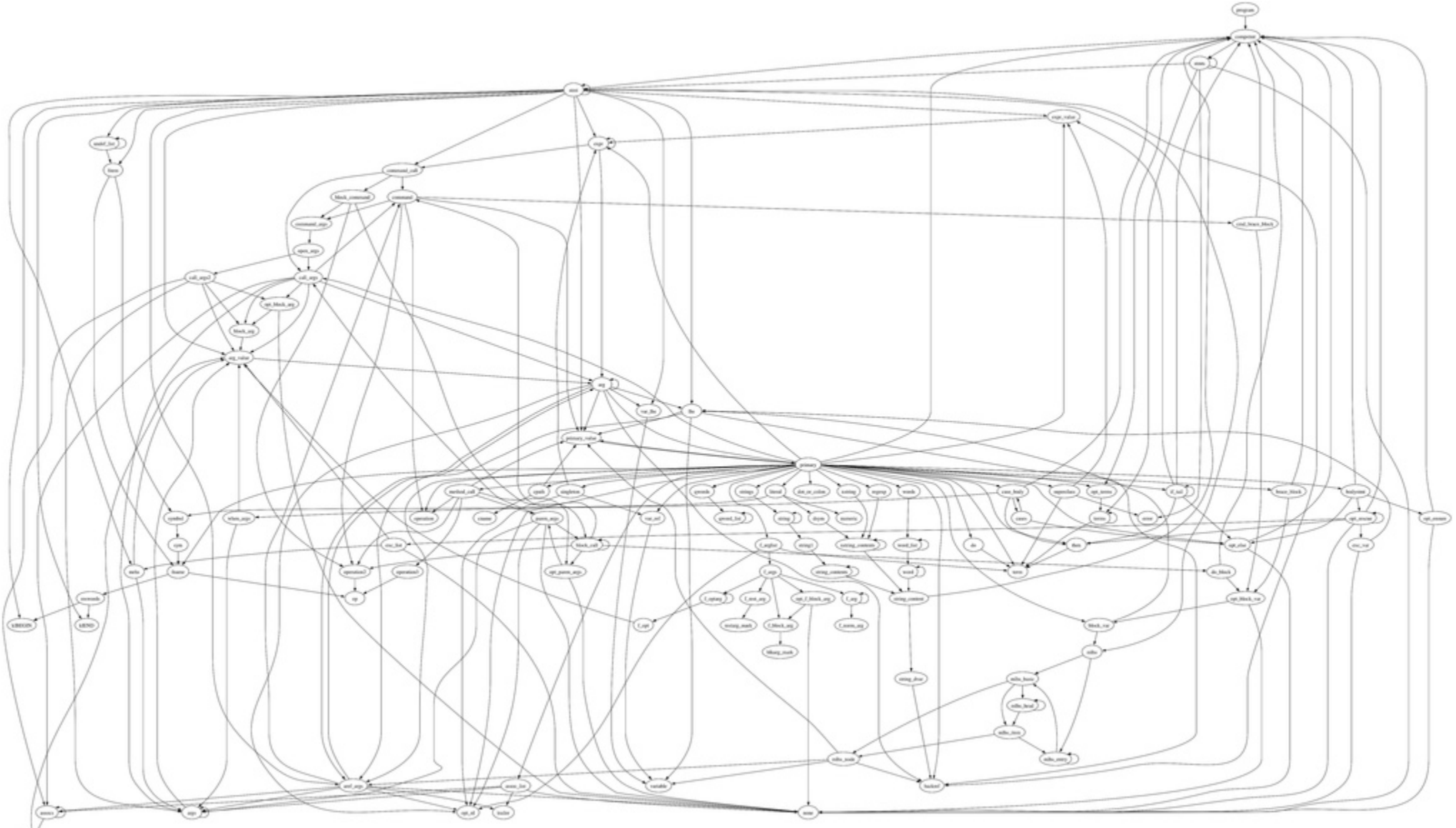
Paradigma de Objetos: C. Accidental

- Consistencia/Homogeneidad
  - Cantidad de casos especiales
- Sintaxis
  - Simple vs. Compleja
  - Expresividad
- Madurez
  - Años de uso
  - Cantidad de usuarios
- Herramientas
  - Representacion concreta vs. Representación textual
  - Contextualización (Debugger)

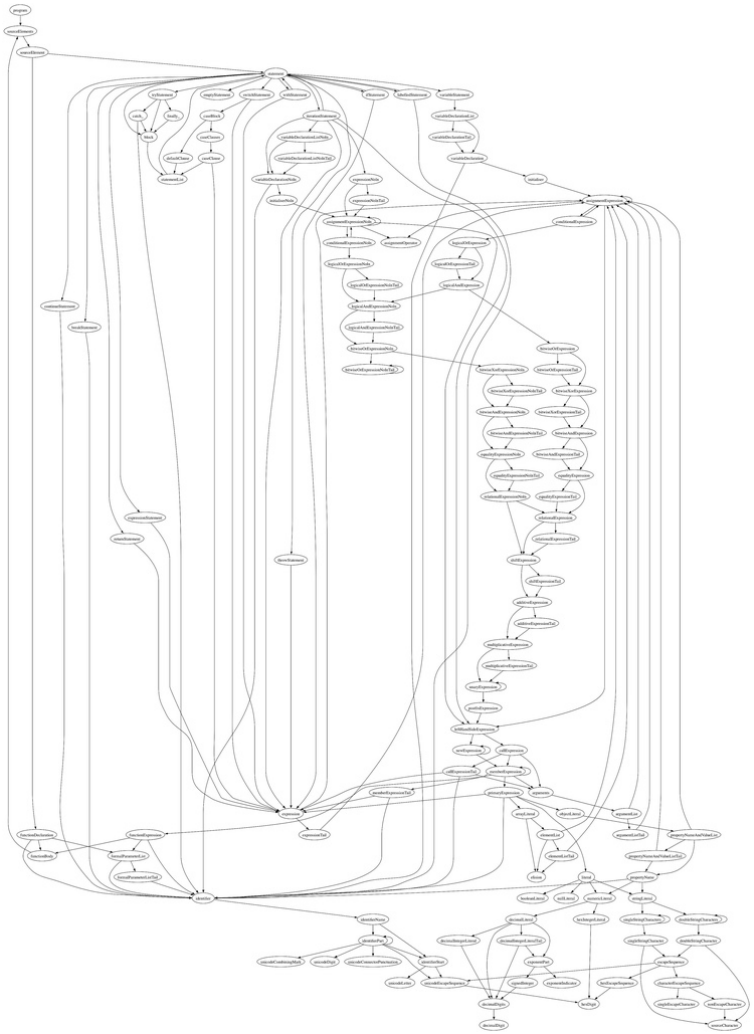
Sintaxis Smalltalk



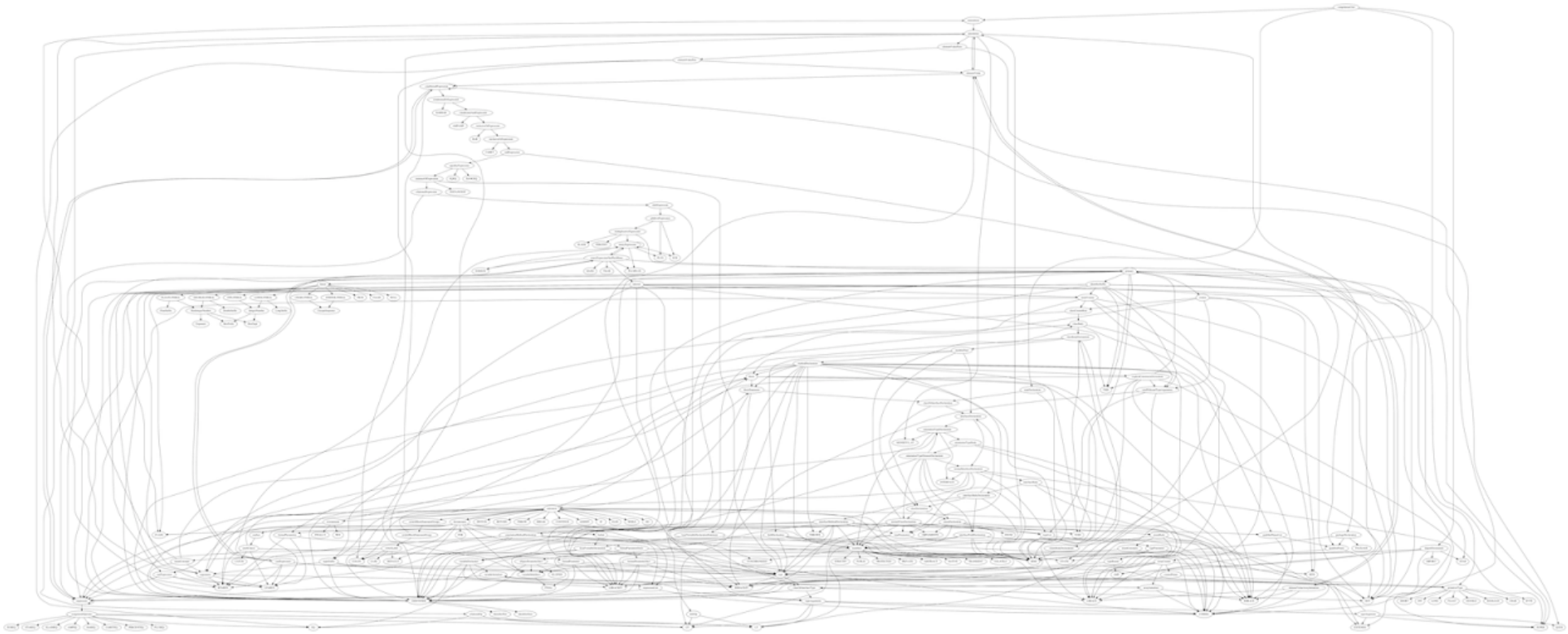
Sintaxis Ruby



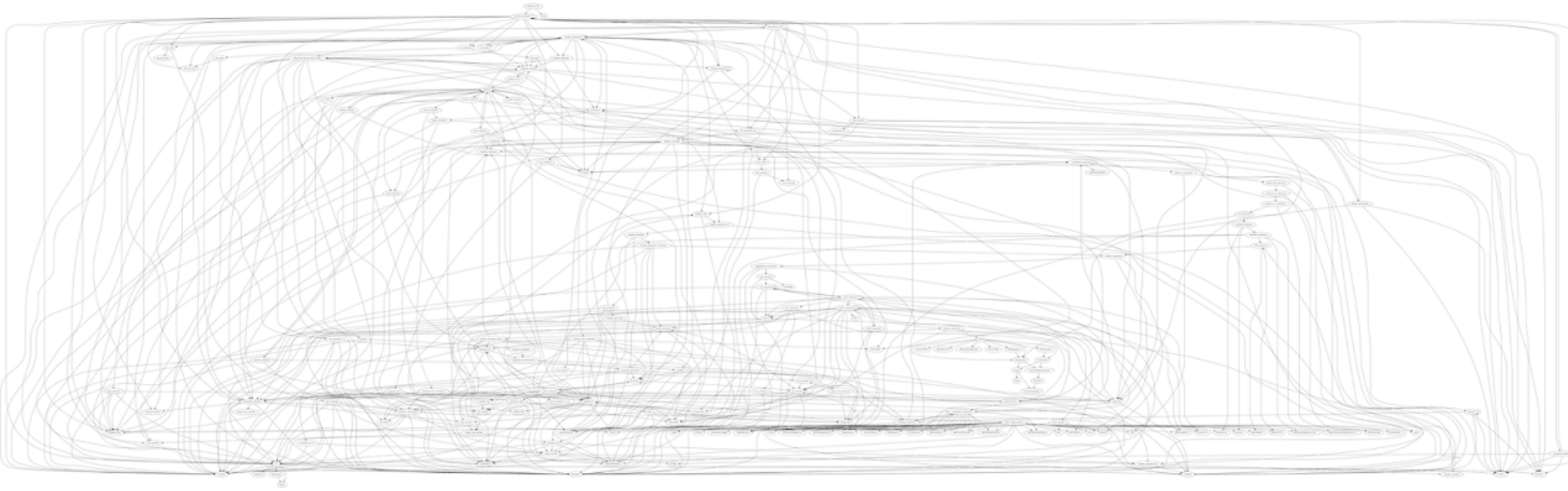
Sintaxis Javascript



Sintaxis Java



Sintaxis C++



Paradigma de Objetos

- Paradigma:
  - Definición minimalista
    - Conjunto de axiomas básicos mínimos
    - Pensar mucho antes de agregar un nuevo axioma
  - Puro
    - No traer conceptos de otros paradigmas, crearlos en base al paradigma
  - Actitud “Fundamentalista”
    - No salir de nuestra actitud minimalista/purista
- Liberarnos del lenguaje
- Liberarnos de otros paradigmas

Paradigma de Objetos

- Programa: **Objetos** que **Colaboran** entre si enviándose **mensajes**

Paradigma de Objetos

- **Objeto: Representación de un ente del dominio de problema**
  - No es código + datos! (error de definición)
  - **Ente: Cualquier cosa que podamos observar, hablar sobre, etc.**
  - La **esencia** del ente es modelado por los **mensajes** que el objeto sabe responder

Paradigma de Objetos

- **Mensaje: Especificación sobre QUE puede hacer un objeto**
  - Un Mensaje es un objeto!
  - Por lo tanto representa un ente de la realidad, pero del dominio de la “comunicación”
  - Se debería poder decir **qué** representa un objeto a partir de los mensajes que sabe responder

Paradigma de Objetos

- **Colaboración: Hecho por el cual dos objetos se comunican por medio de un mensaje**
  - En toda colaboración existe:
    - Un emisor del mensaje
    - Un receptor del mensaje
    - Un conjunto de objetos que forman parte del mensaje (colaboradores externos o parámetros)
    - Una respuesta

Paradigma de Objetos

- Características de las colaboraciones:
  - Dirigida (no broadcast)
  - Son sincrónicas, el emisor no continúa hasta que obtenga una respuesta del receptor
  - El receptor desconoce al emisor → su reacción será siempre la misma no importa quién le envía el mensaje
  - Siempre hay respuesta
- ¿Qué pasa si se cuestiona alguna de estas características?
  - Subjectivity (context aware…)

Paradigma de Objetos

- **Método**
  - Objeto que representa un conjunto de colaboraciones
  - Es evaluado como el resultado de la recepción de un mensaje por parte de un objeto
  - El método se busca utilizando el algoritmo de *Method Lookup*
  - Mismo mensaje principal que un programa: *execute* o *value*

Paradigma de Objetos

- **Relación de Conocimiento**
  - Es la única relación que existe entre objetos
  - Es la que permite que un objeto colabore con otro

Paradigma de Objetos

- **Colaborador/Variable**
  - Nombre contextual que se le asigna a una relación de conocimiento
  - Implica que el objeto conocido es llamado, en el contexto del "*conocedor*", de acuerdo a dicho nombre
  - No tiene ninguna implicación respecto del lugar que ocupa en memoria, cuánto ocupa, etc.

Paradigma de Objetos

- **Pseudo Variable “self” o “this”**
  - Nombre que referencia al objeto que está evaluando el método

- Características de las pseudo-variables:
  - No deben ser definidas
  - No pueden ser asignadas