Construction (et exploitation) d'un référentiel de types de tâches d'apprentissage de la programmation

Sébastien Jolivet, Patrick Wang, Eva Dechaux, Anne-Claire Gobard

UNIVERSITÉ DE GENÈVE
INSTITUT UNIVERSITAIRE DE FORMATION DES ENSEIGNANTS

tecfa
Technologies de formation et apprentissage

LYCÉE
Alfred Kastler
Cergy-Pontoise

hep
haute école pédagogique vaud

Plan global

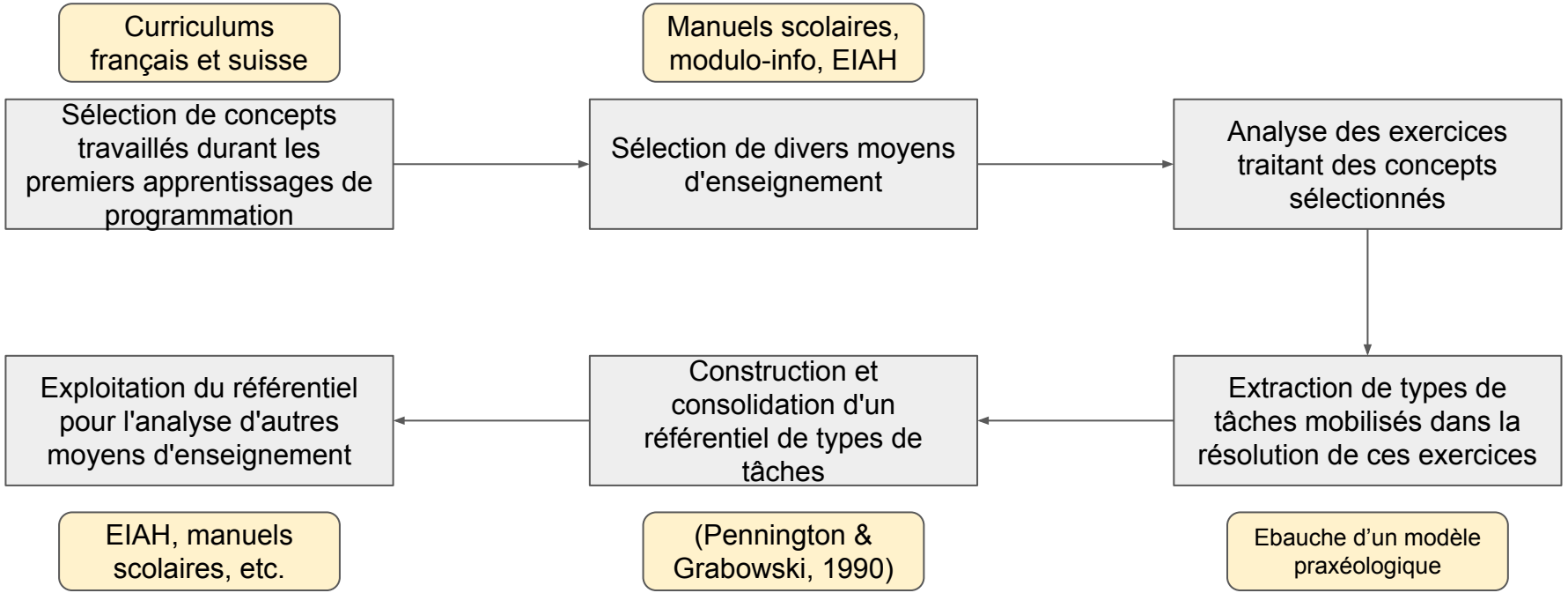
- Contexte ayant conduit à la production du référentiel / modèle
- Processus de construction du référentiel
- Quels éléments structurants ?
- Quels choix de représentation / formalisation du référentiel ?
- Quelles exploitations étaient envisagées lors de la conception du référentiel ?
- Ouvertures

Contexte de production / exploitation du référentiel

Enseignement de l'informatique :
→ En France, SNT / NSI
→ En Suisse, informatique au secondaire 1 (12-15 ans) et 2 (15-18 ans)
Besoin de répondre à la question : « (apprendre à) programmer, c'est quoi ? »

IUFE / HEP : Instituts de formation des futur·e·s enseignant·e·s d'informatique
Besoin de répondre à la question : « enseigner la programmation, c'est quoi ? »

Processus simplifié de construction du référentiel



A l'occasion de plusieurs étapes interviennent : l'expertise des auteurs ; les travaux de la didactique de l'informatique ; ouvrages de référence en informatique

Eléments structurants du référentiel

Fondement théorique : le modèle praxéologique (description de toute activité humaine, donc aussi celle de programmer, à l'aide des 4 T) dans le cadre d'une approche anthropologique (par exemple à la différence d'une approche qui serait orientée par la représentation informatique des objets, par les langages, etc.)

Objectif : décrire des praxéologies complètes avec le bloc praxique et le bloc du logos

Actuellement : la description est centrée sur le bloc praxique avec

- une description, structurée autour d'objets considérés comme fondamentaux, de types de tâches et des différents types d'activités
- l'identification d'ingrédients de la technique de ces types de tâches, ingrédients qui sont eux-même des types de tâches
- un logos qui est largement absent au delà de l'identification des objets fondamentaux

Eléments structurants du référentiel

Programming subtasks	Basic processes	Knowledge domains	Mental representations	External representations
Understanding the problem	COMPOSITION / COMPREHENSION	Domain knowledge (e.g., statistics, banking)	Situation model	Requirements document Specifications document
Design		Design strategies Programming Algorithms and methods Design language	Solution model Plan representation	Design document
Coding		Programming language Programming conventions	Program representation	Code
Maintenance		All knowledge domains Debugging, testing strategies Frequent kinds of error	All representations	All documents

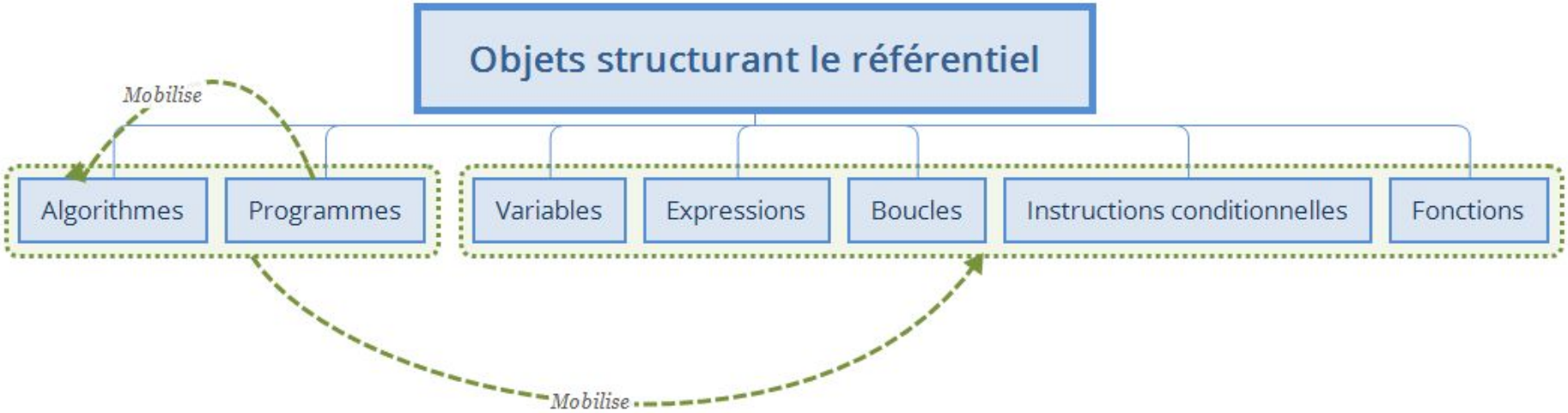
Figure 1: The tasks of programming.

Pennington & Grabowski, 1990

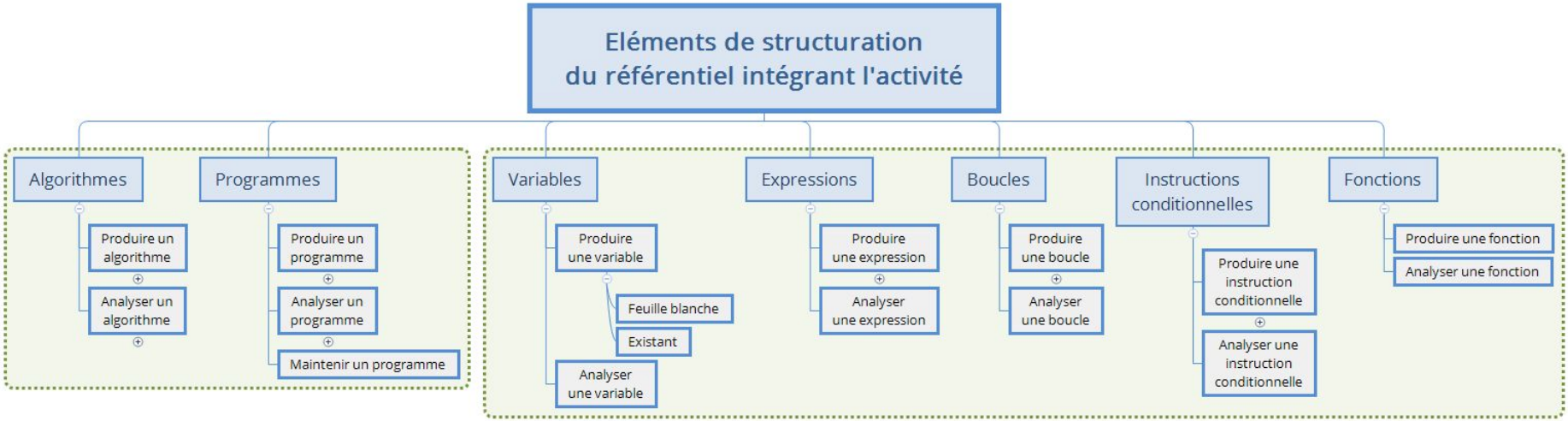
Objets : algorithme programme

Activités : produire analyser

Eléments structurants du référentiel, selon objets



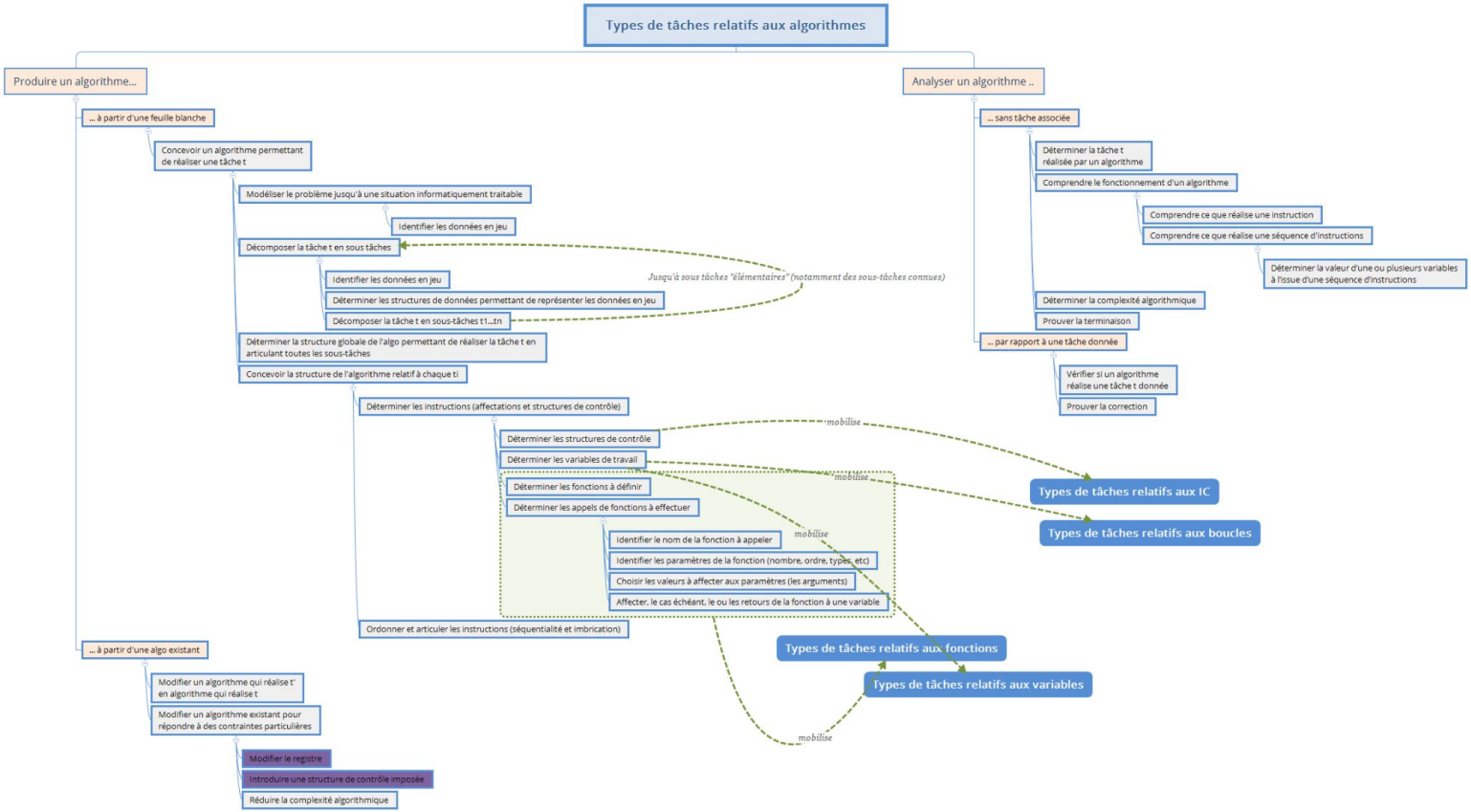
Eléments structurants du référentiel en intégrant l'activité



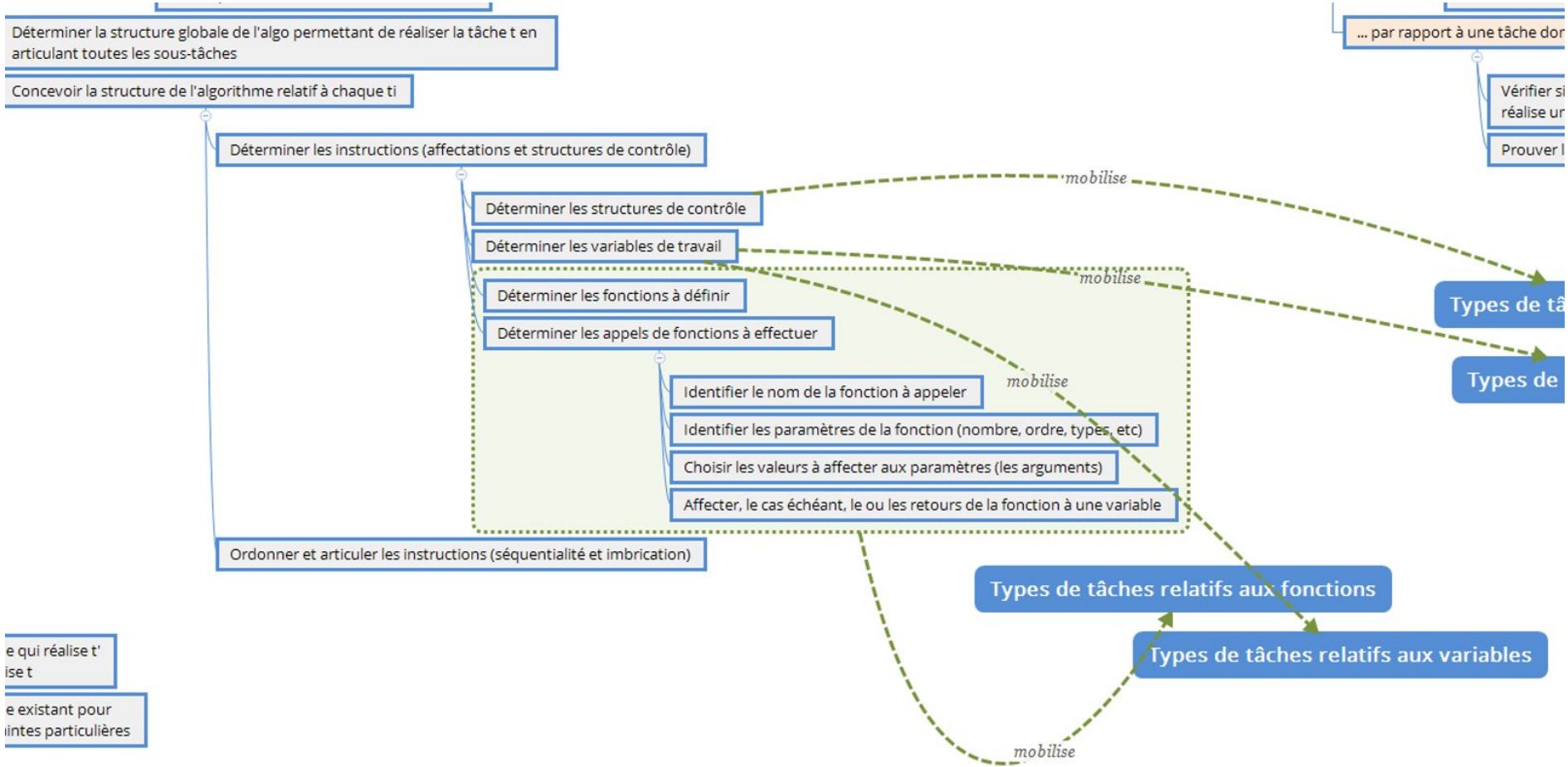
Représentation / formalisation du référentiel

Choix de représentation / formalisation du référentiel de types de tâches

- Ad-hoc sous forme d'arbre
- Identifications de premières relations entre les objets
  - enfant dans l'arbre -> élément d'une technique du type de tâches
  - relations entre les concepts (un type de tâches de X mobilise des types de tâches de Y -> mettre un exemple)



Types de tâches relatifs aux algorithmes



Types de tâches relatifs aux algorithmes (zoom)

Représentation / formalisation du référentiel

Limites et difficultés liées à ce mode de représentation

la représentation en arbre ne permet que la relation “parents - enfants” (pour nous types de tâches - type de tâches ingrédient de la technique) alors que l’on souhaiterait modéliser au moins deux relations :

  - types de tâches - type de tâches ingrédient de la technique
  - type de tâches - sous-type de tâches
- la représentation en arbre peut donner une impression de séquentialité systématique (lecture de haut en bas) alors que ce n’est pas systématiquement le cas
- l’introduction des éléments relatifs au logos est particulièrement complexe avec cette représentation

Perspectives

- Compléter la représentation actuelle avec des générateurs de types de tâches
- Intégrer la problématique du logos

  In fine un élément du référentiel est une praxéologie complète [Type de tâches ; Technique ; Logos]

- Aller vers une ontologie

Autour du référentiel

Présentation du référentiel

→ (EIAH23)
→ (DIDAPRO10, https://hal.science/hal-04482123/)

Analyses de ressources:
→ (APIMU'23, https://hal.science/hal-04144212/) : Py-rates, AlgoPython, Citizen Code Python
→ (DIDAPRO10, https://hal.science/hal-04482123/) : énoncés d'exercices

Référentiel mis en ligne (CC NC-BY-SA) : https://link.infini.fr/ref_prog
→ *read-only*, mais volonté d'avoir un hébergement/format permettant la co-construction, les modifications, le suivi de versions, les annotations, …

Les éléments représentés, leur structuration et la représentation associée :

- n'ancrent pas le référentiel dans un usage ou une institution spécifique
- permettent une forte évolutivité

Ouvertures, pistes, libre expression… (1/2)

Travail en évolution :

→ consolidation de la structure

→ couverture plus large en considérant d'autres notions

→ adaptation pour des usages différents

Modalités d'exploitation :

→ analyses de ressources / génération de ressources

→ est-ce que ce référentiel est un outil (ou n'est-il pas) pour la formation des enseignants ou pour l'enseignement de la programmation ?

→ mises en regards avec le travail de La Réunion

Ouvertures, pistes, libre expression… (2/2)

Vers les générateurs de types de tâches, modèle T4TEL (Chaachoua & Bessot).

Retour sur une différence entre didactique de l'informatique et didactique des mathématiques : éléments pour fonder le choix des variables et de leurs valeurs.

Verbe action	Complément fixe	V1 : type de boucle	V2 : nb instruction dans le corps	V3 : nb de répétitions	V4 : itérateur dans le corps de la boucle
Concevoir	une boucle	Bornée	1	<10	Oui
		Non bornée	>1	10<...<100	Non
				<100	
				Sans objet	