Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ proj. no. 2020-1-PT01-KA203-078646
SusTrainable - Promoting Sustainability as a Fundamental Driver in Software Development Training and Education

VERIFIED SOFTWARE IN VIRTUAL ENVIRONMENTS

(DEVELOPMENT OF CORRECT SOFTWARE WITH FORMAL METHODS)

Štefan Korečko

Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice

https://kpi.fei.tuke.sk/en/person/stefan-korecko

stefan.korecko@tuke.sk

Contents

- Verified Software Development
  - verification vs validation
- Validation in Virtual Environments
  - relation to sustainability
- Cleaning Robot Case Study
  - assignment
  - technologies
  - development process
  - verified controller development
  - building virtual environment for validation

Lecture

Practice (exercise)

This presentation

https://bit.ly/VsInVePrez

Verified Software Development

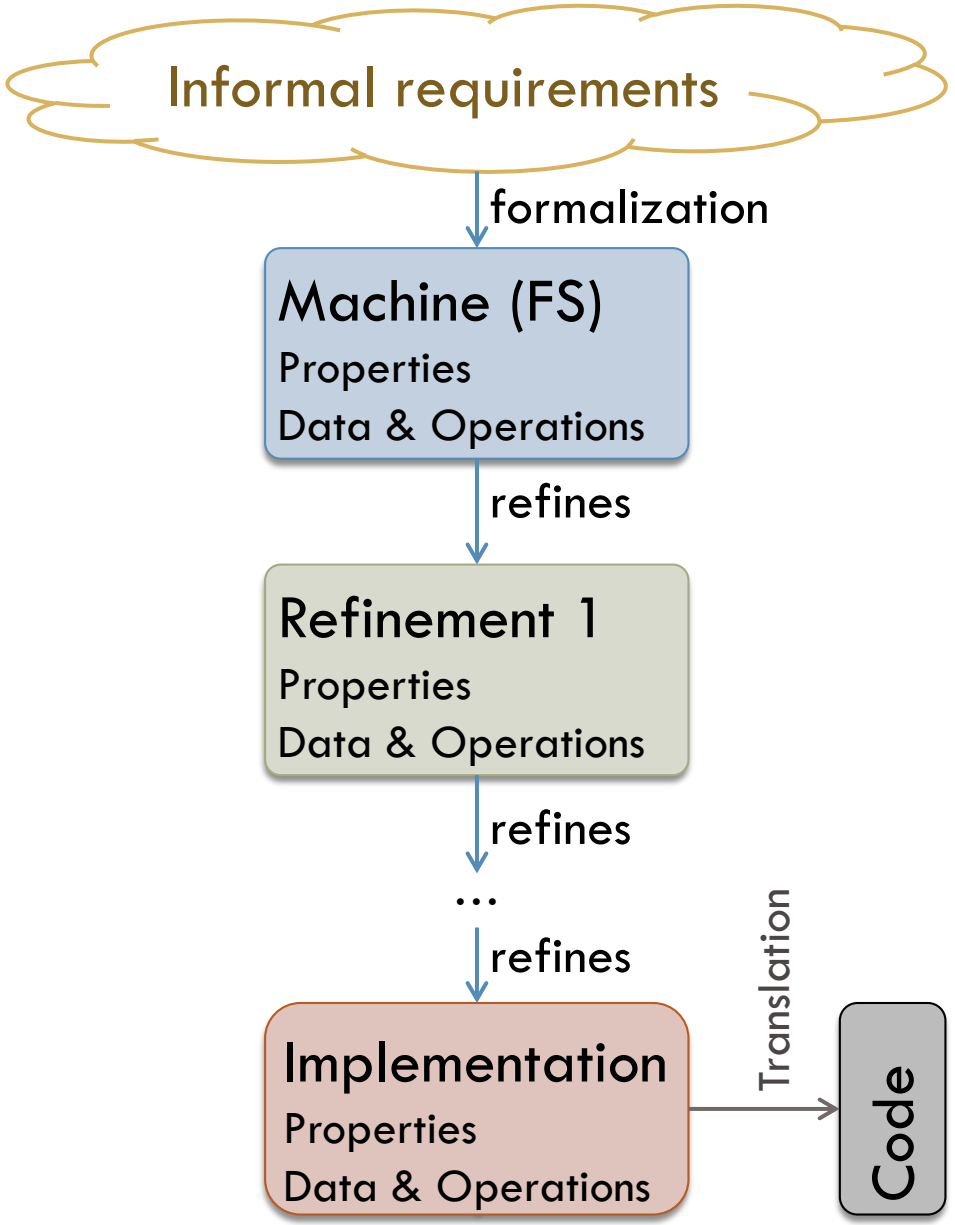
Verified software development

- Verified sw development
  - = sw development with formal methods (FM)
- In verified sw development
  - correctness is proved against formally specified requirements (verification)
  - no way to check formally whether the requirements are correct (validation)
- FM allow us to have sw specifications for verification early
- How to prepare environments (contexts) for validation early, too?

How verified sw development works

- E.g. in B-Method

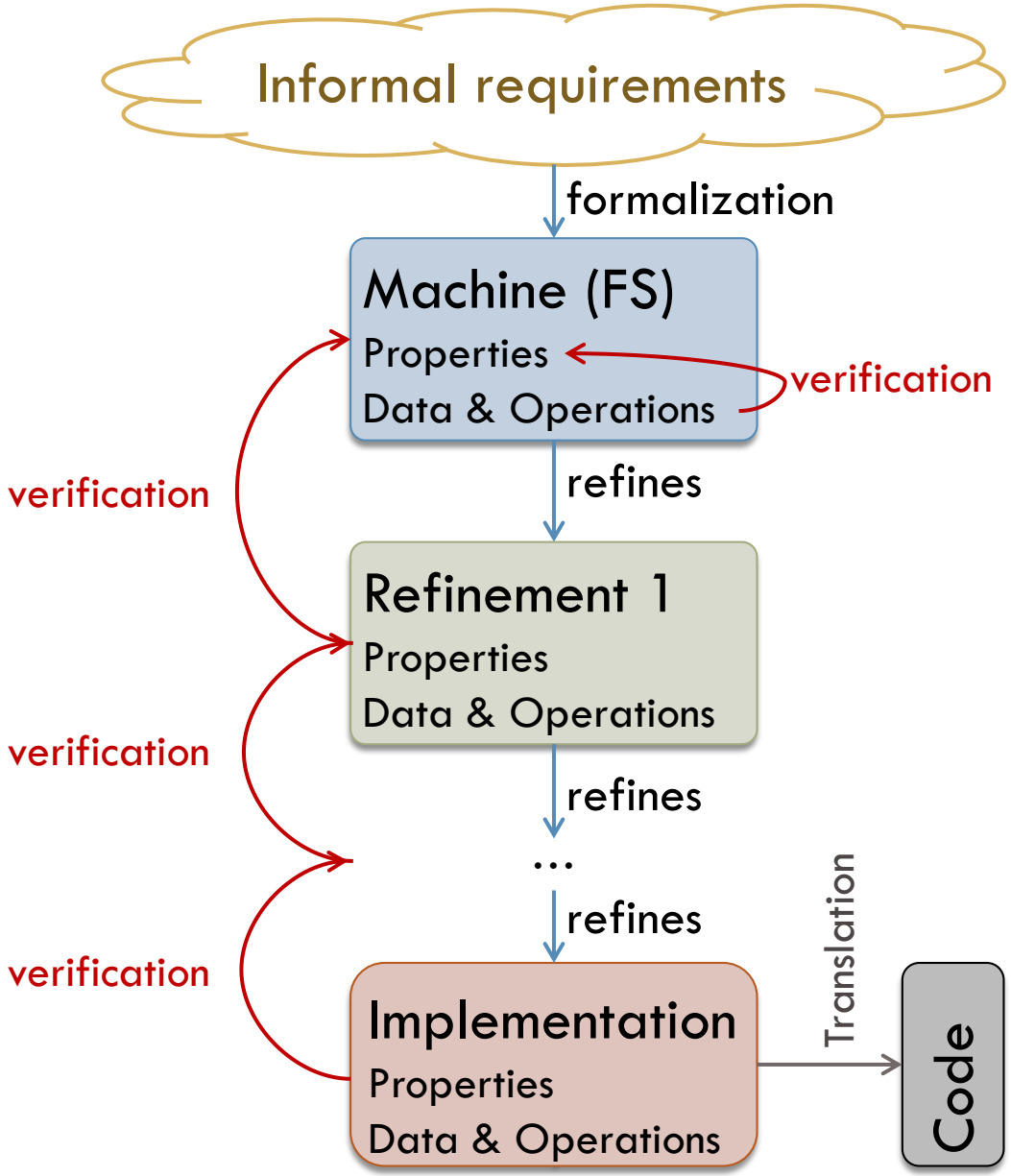
FS = Formal Specification



Verified sw development: verification

Verification

- Checks whether the system has the desired properties
  - By theorem proving or model checking
  - The **properties have to be formalized**

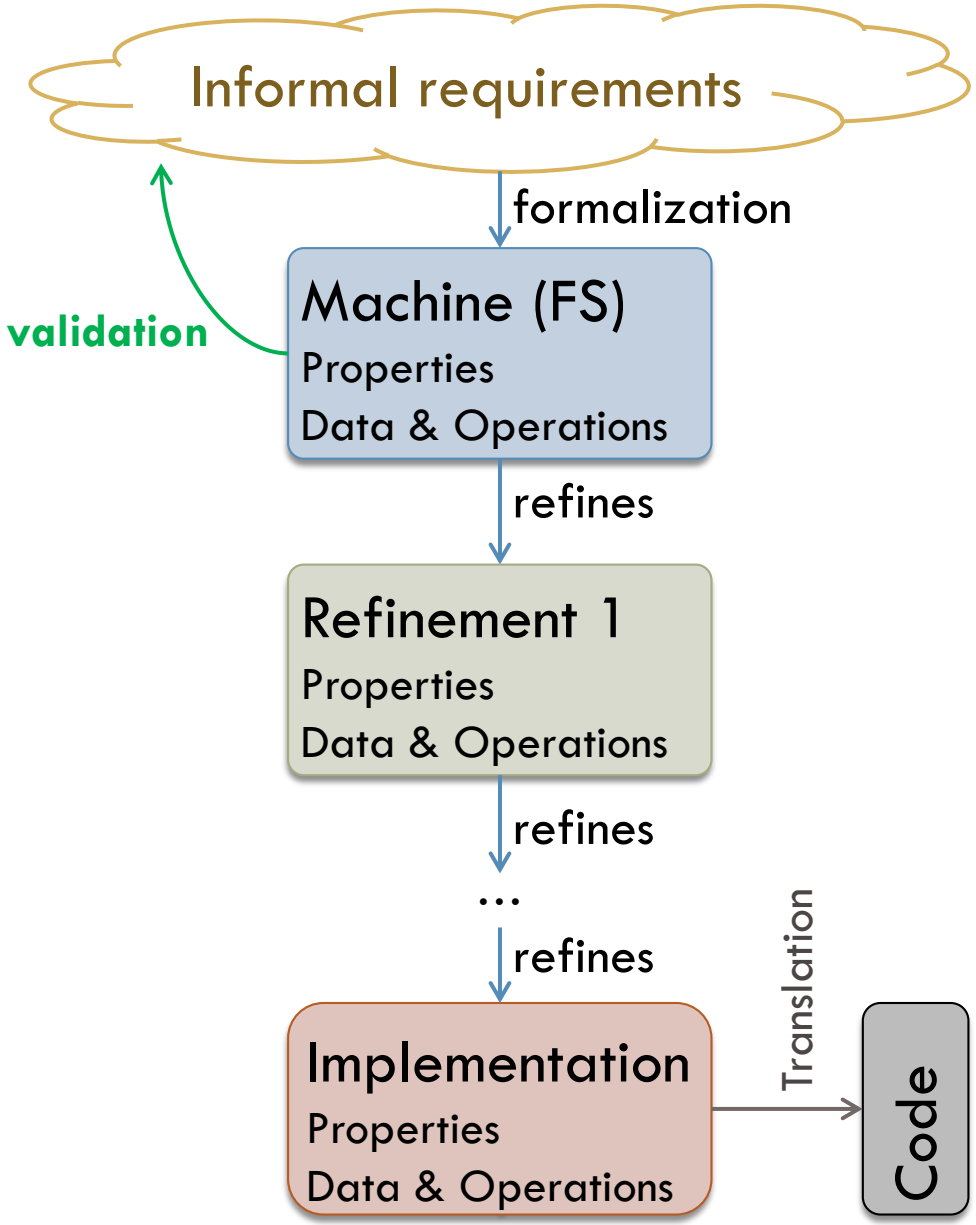


Verified sw development: validation

Validation

- Checks whether the formalized properties capture the informal requirements
  - Manual check: properties vs. requirements
  - Animation
    - "running" the specification

If the organization has not built the right system (validation), no amount of building the system right (verification) can overcome that error."
(J. Bowen and M. Hinchey, 2006)



Validation in Virtual Environments

Validation in virtual environments

- To obtain environments (context) for validation
  - use virtual environments
    - Web-based (online)
    - Shared
    - Capturing only those aspects of the real context that are important for the validation
- Why now?
  - Easy access to virtual environments
    - Can run directly in a web browser
      - Computers powerful enough
      - Software frameworks available
  - VR hardware (VR headsets)
    - Affordable: about 500 Eur/unit
    - Full immersion possible
    - Real-time motion tracking

Relation to Sustainability

- It contributes to the environmental and economic sustainability.

- Virtual prototype
  - instead of a real one
- Testers (validators) may access the system online
  - No need to travel

- Re-usability of assets when building virtual environments

Cleaning Robot Case Study

Assignment & Technologies

Development Process Overview

Verified Controller Development

Building Virtual Environment for Validation

Case Study Assignment I

- Assignment:
  Develop
  - a verified software controller prototype and
  - a virtual environment to validate the prototype

  for an autonomous cleaning robot.

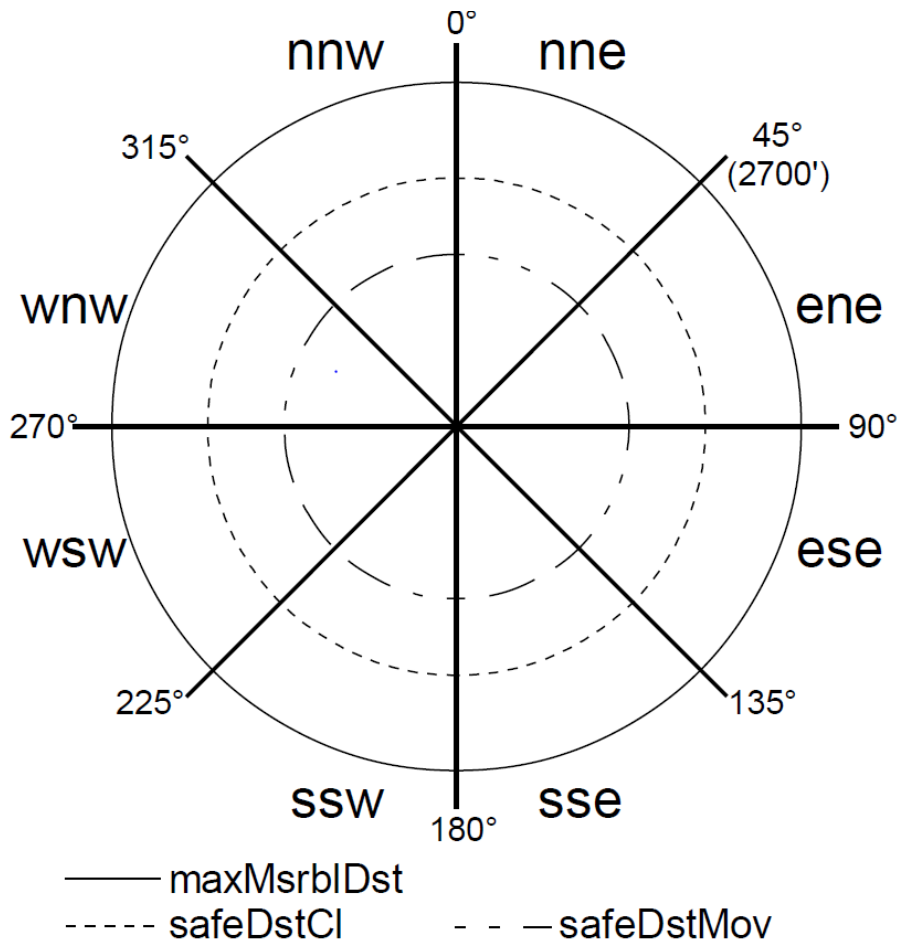
- Try the result now
  - https://bit.ly/VsInVeC

Case Study Assignment II

The autonomous cleaning robot

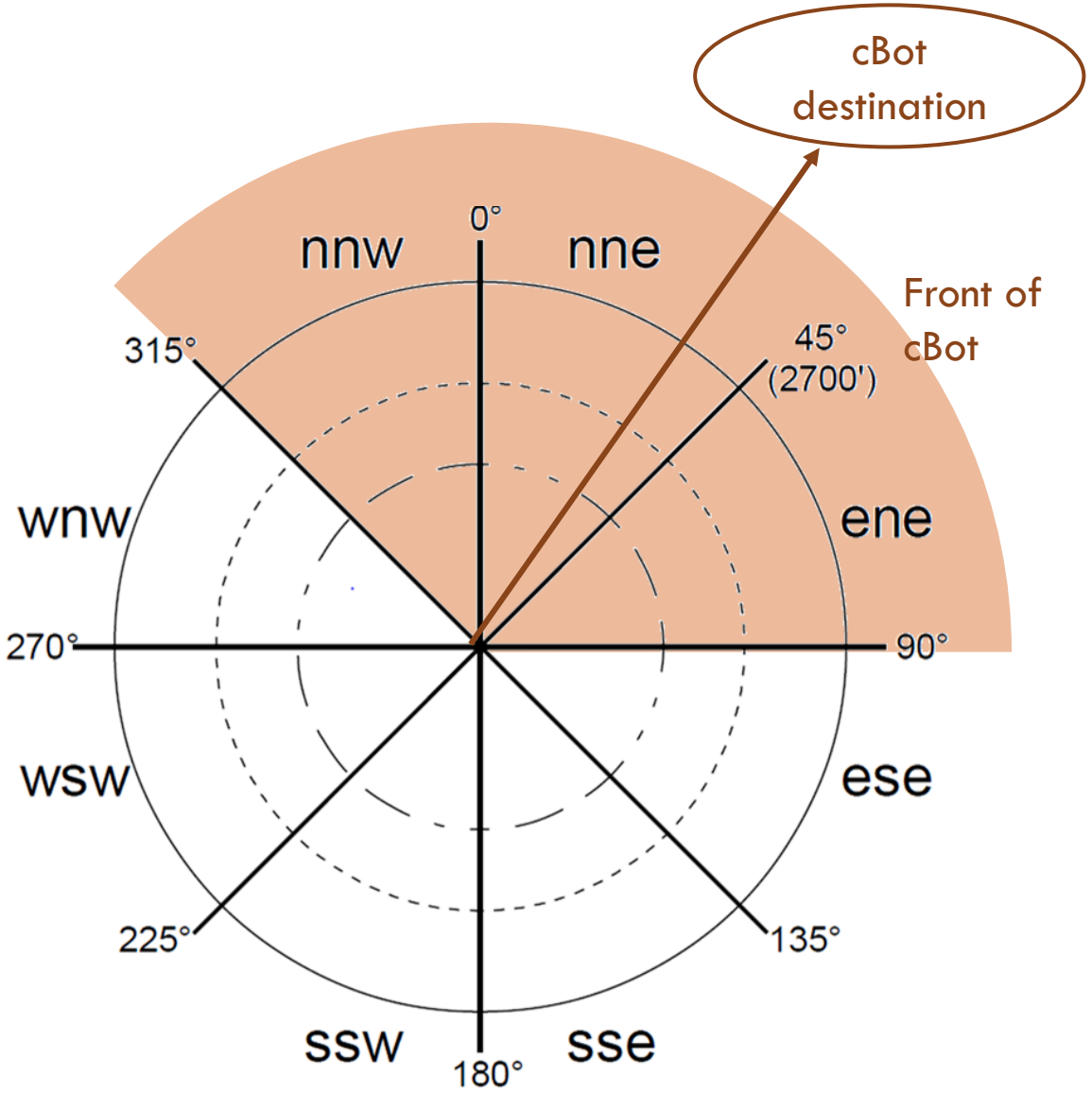
- Task: To clean several locations ina public space.
- Has a sensor array to detect persons nearby.

- Safety properties
  - The cleaning cannot start or continue if anyone gets as close or closer to the robot as safeDstCl.
  - The robot cannot move if anyone gets as close or closer to its front as safeDstMov.



Case Study Assignment II: What is front?

- cBot cannot move if anyone gets as close or closer to its front as safeDstMov.
- What is front???



Technologies used

- Formal method for sw development
  - B-Method
    - Atelier-B, https://www.atelierb.eu/en/
    - BKPI compiler, https://hron.fei.tuke.sk/~korecko/FMInGamesExp/resources/BKPICompiler.zip
- Virtual environment
  - Web-based virtual reality
    - A-Frame, https://aframe.io/
    - Networked-Aframe, https://github.com/networked-aframe/networked-aframe

Cleaning Robot Case Study

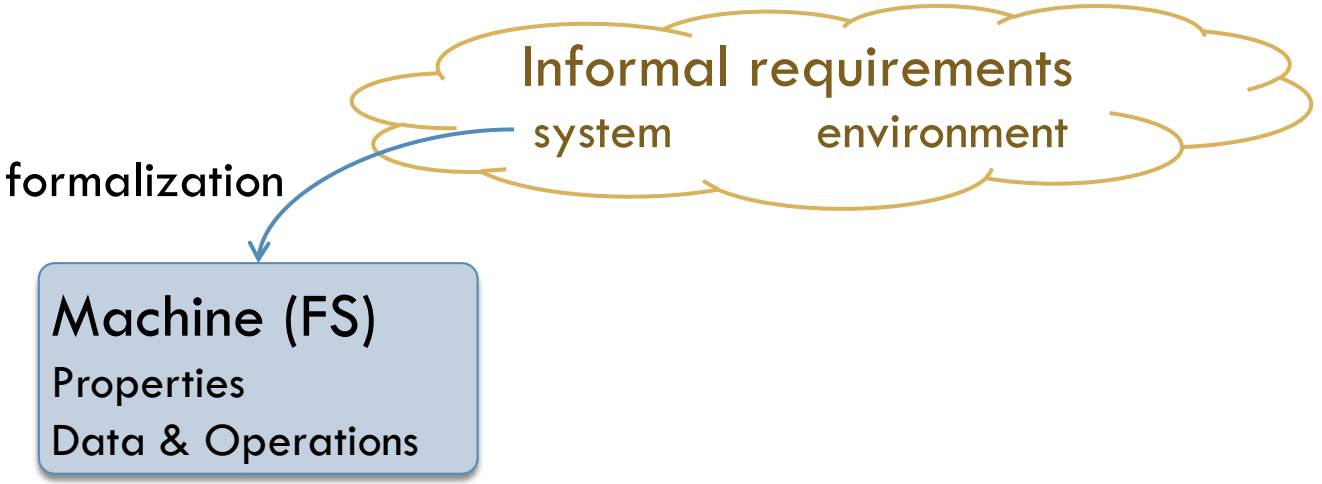
Assignment & Technologies

Development Process Overview

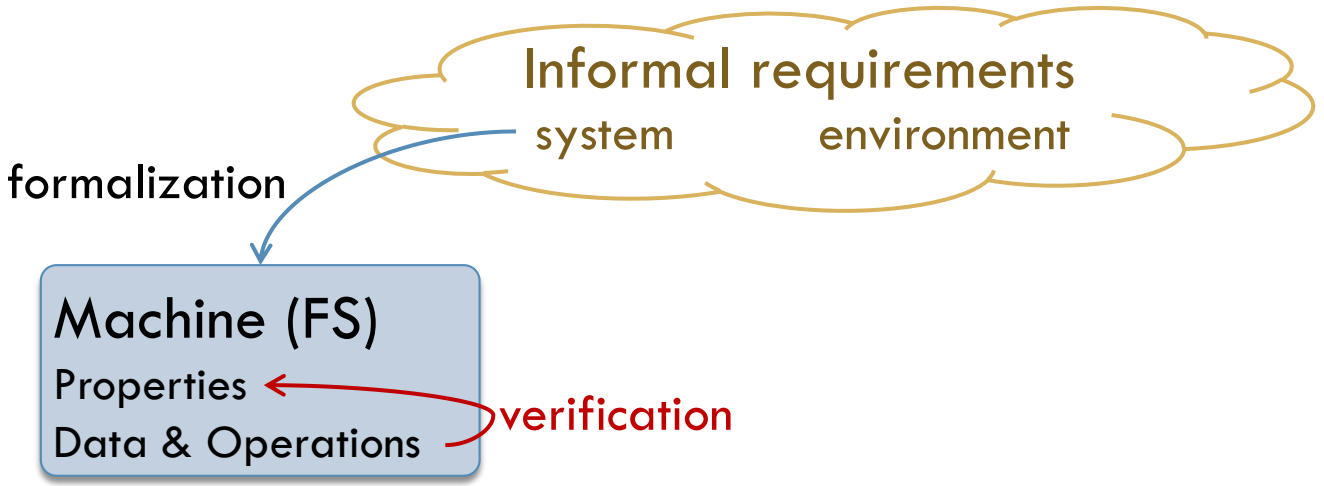
Verified Controller Development

Building Virtual Environment for Validation

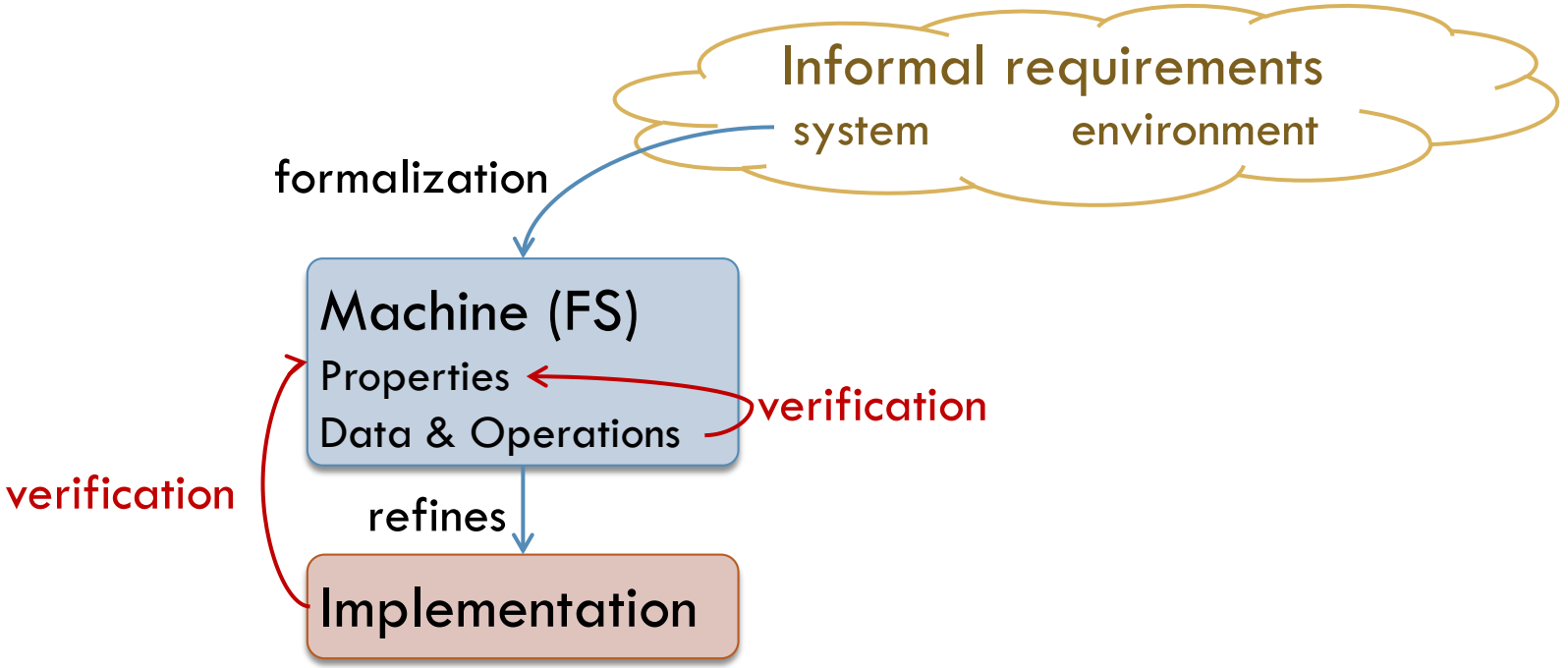
Case study: development process



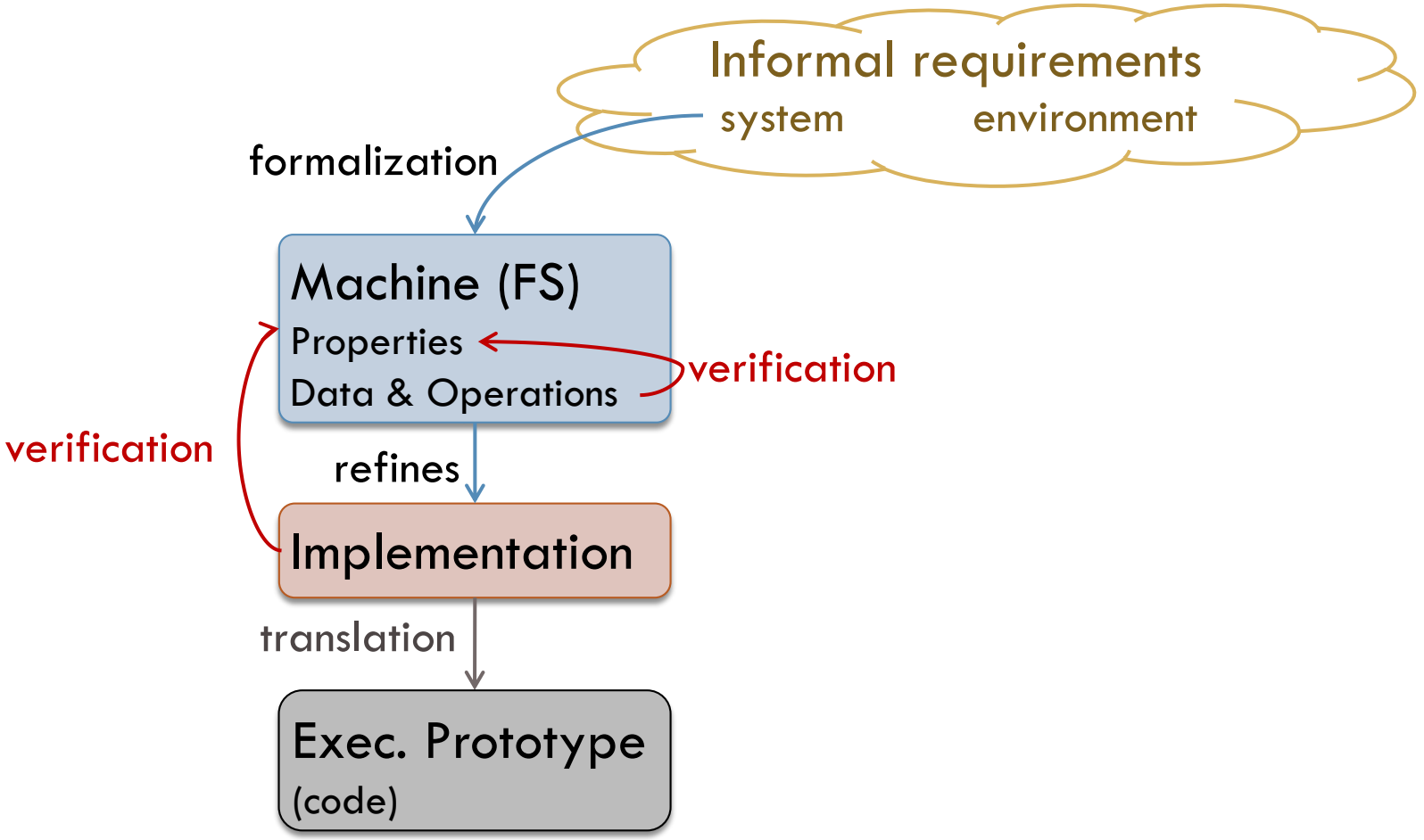
Case study: development process



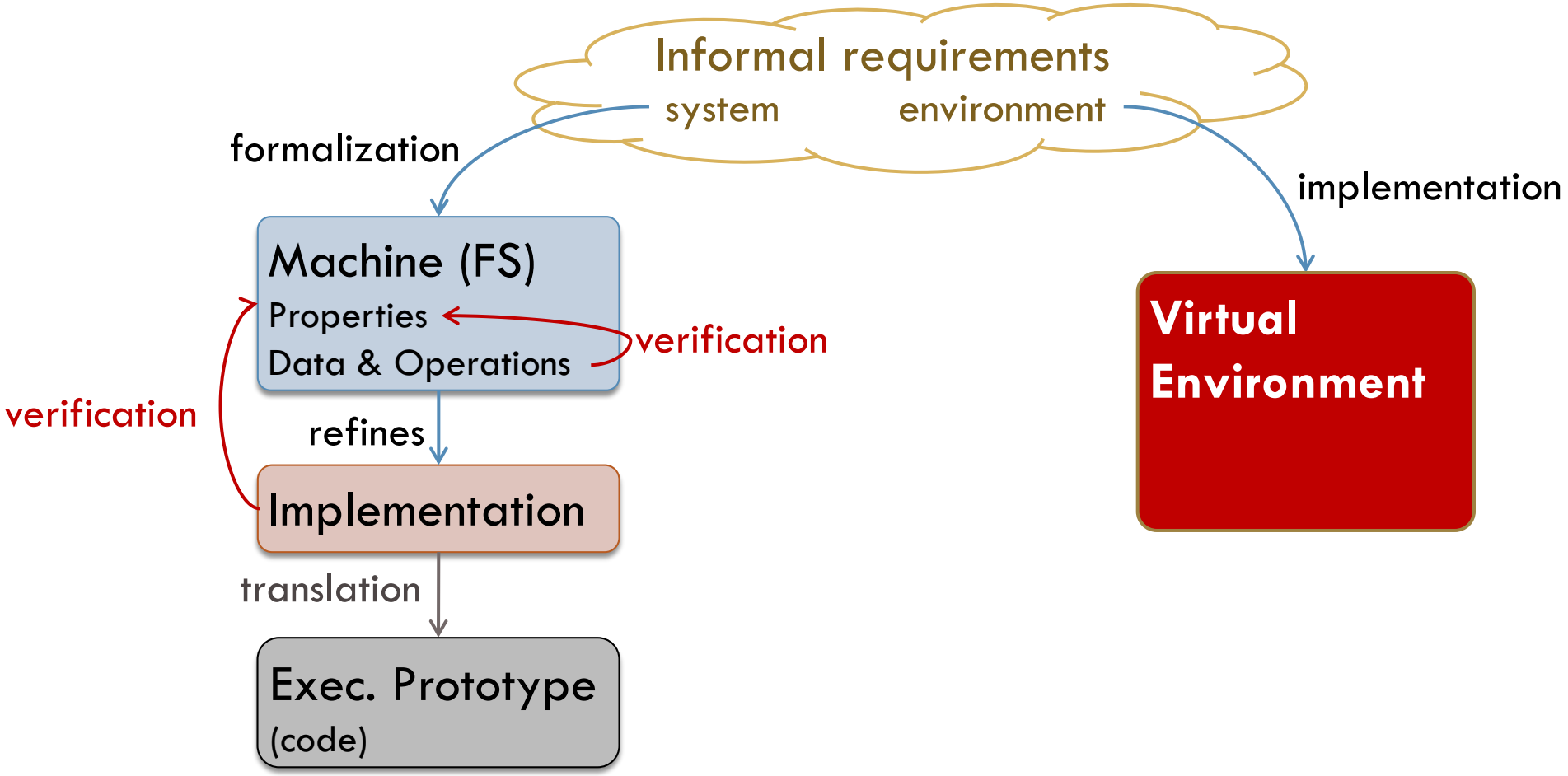
Case study: development process



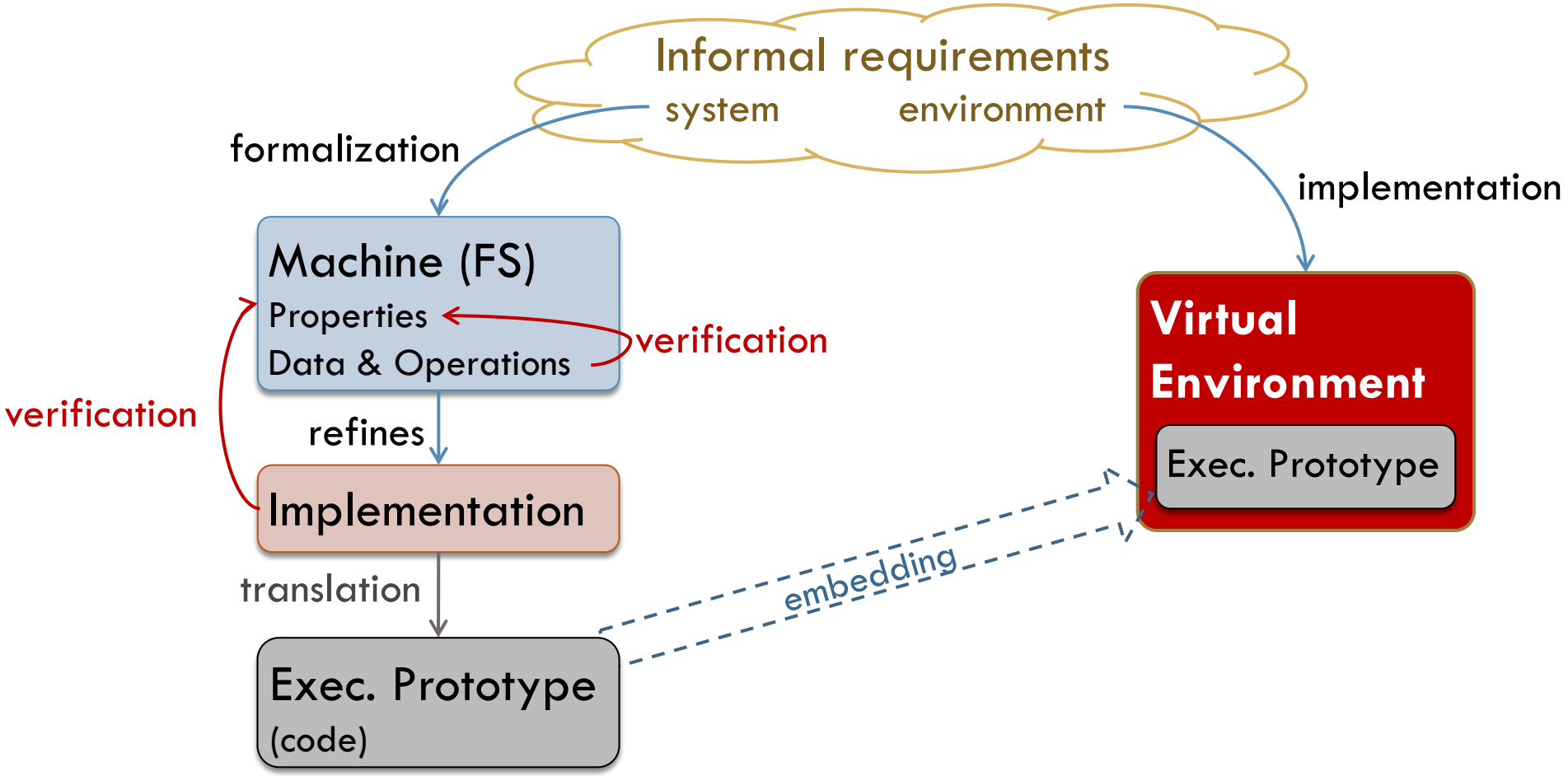
Case study: development process



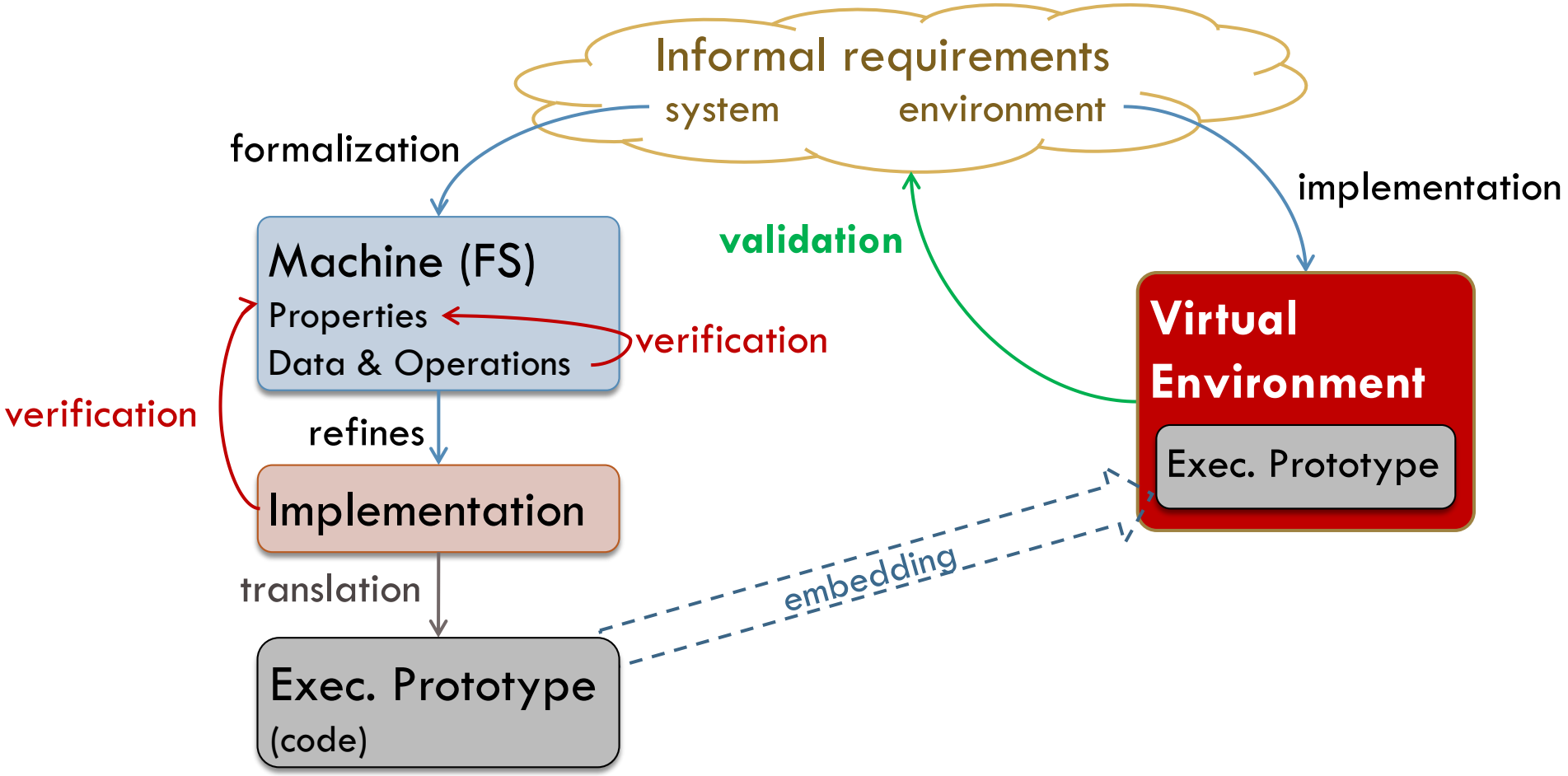
Case study: development process



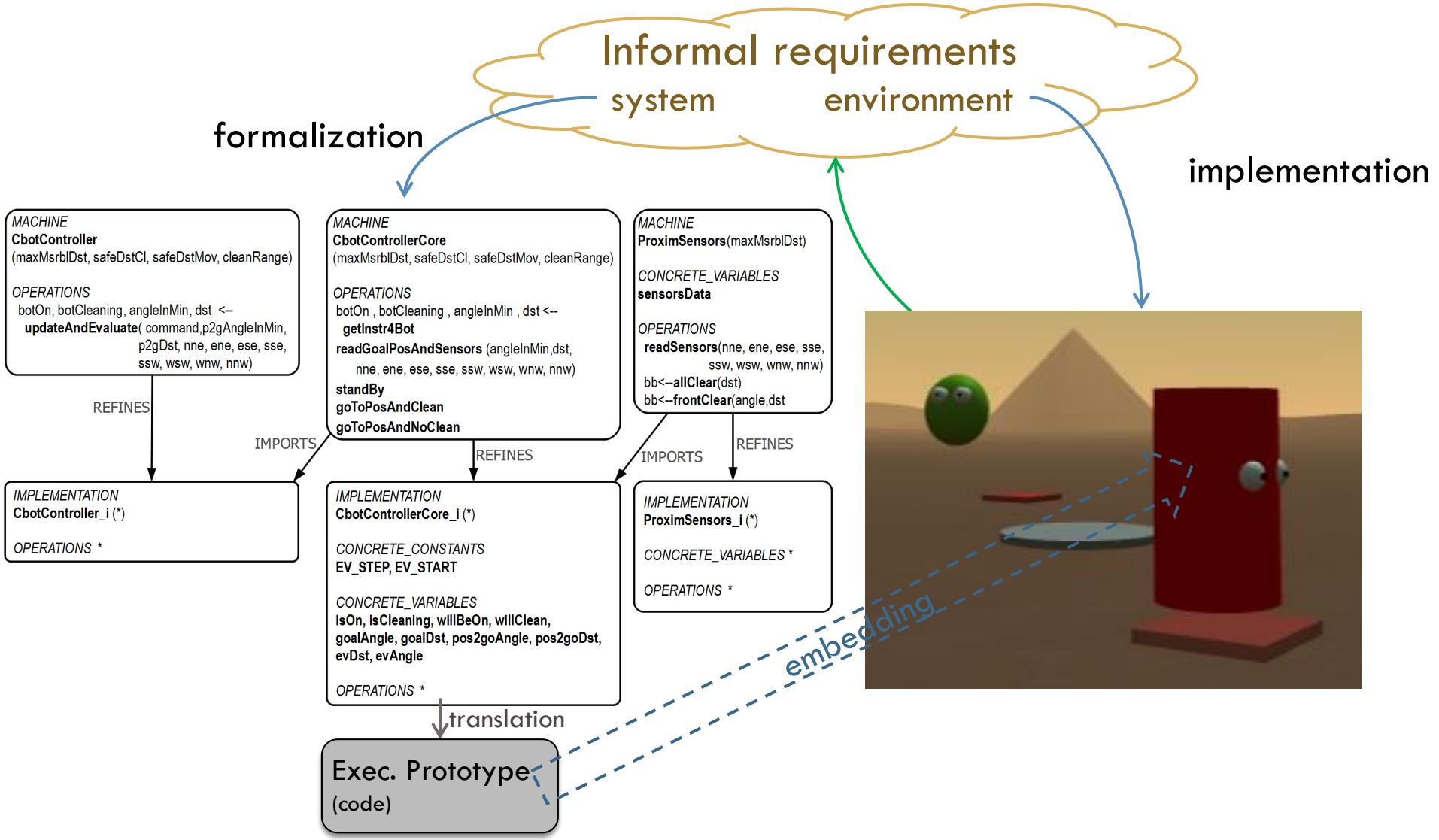
Case study: development process



Case study: development process

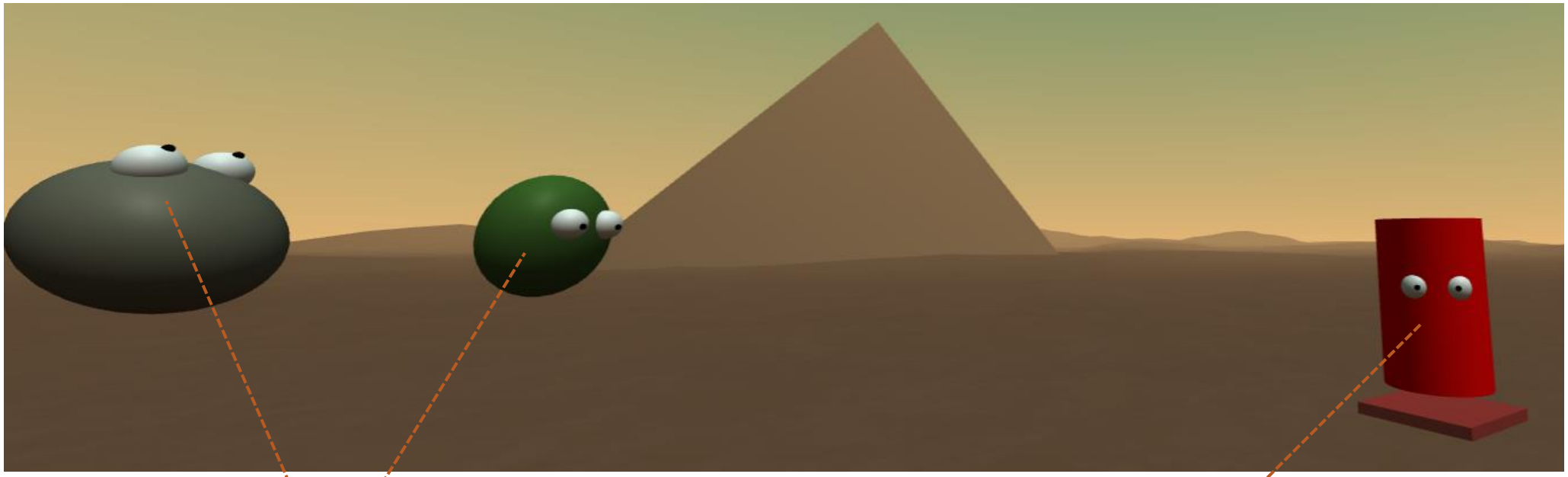


Case study: development process



Case study: final form

- https://lirkis-cbot-simple.glitch.me/
  - Citizen representing a tester (validator)
- https://lirkis-cbot-simple.glitch.me/cbot.html
  - the autonomous cleaning robot with the verified software controller



Cleaning Robot Case Study

Assignment & Technologies

Development Process Overview

Verified Controller Development

Building Virtual Environment for Validation

Verified controller role

- Works within a main controller that controls the whole cBot and its interaction with the virtual environment
  - Implemented in `cbot-master-controller.component.js`

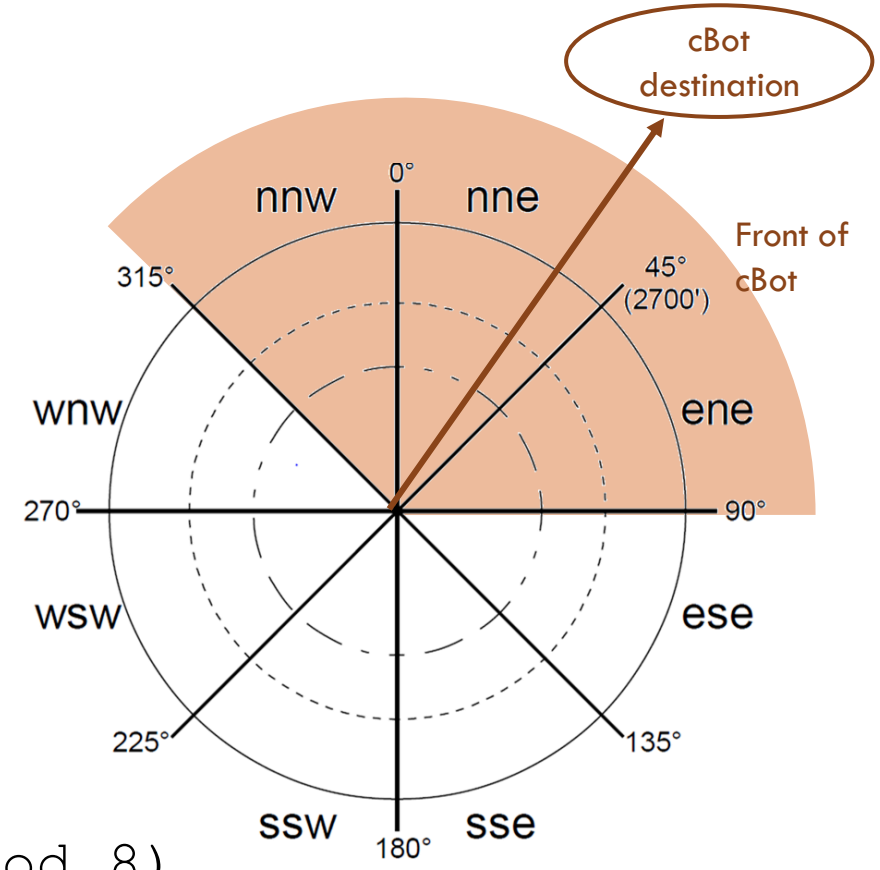
- In each step (frame), the main controller operates like this:
  1. Check what should be done, i.e. clean a position or go to a parking position.
  2. Check position of other citizens (users) in the scene.
  3. **Call the verified controller (function `updateAndEvaluate`) to decide what to do, without endangering other citizens.**
  4. Update the cBot position, state and colors according to the output of the verified controller. Aslo update the dirty positions.

Safety properties formalization

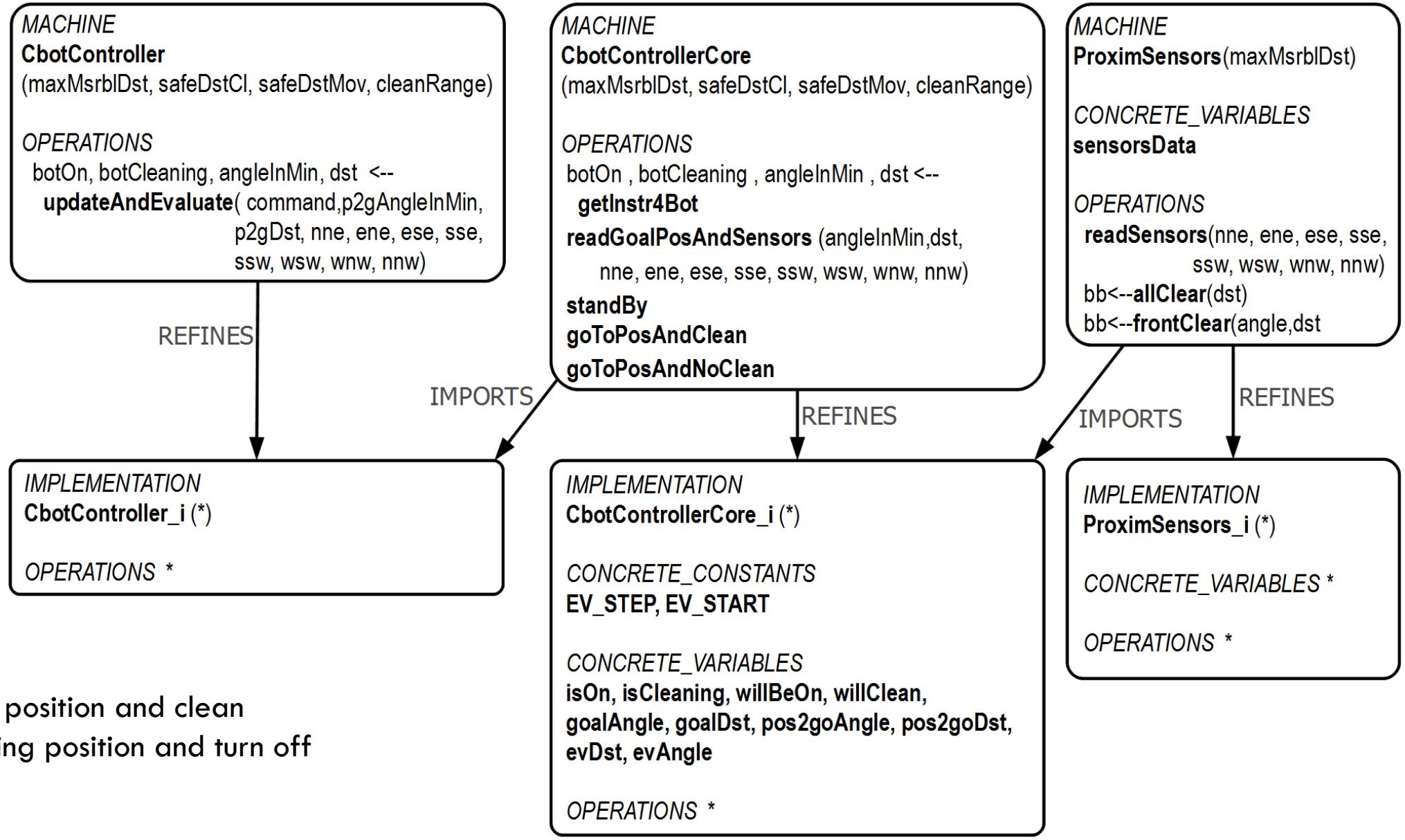
- Specified in `CBotControllerCore_i.imp`

- The cleaning cannot start or continue if anyone gets as close or closer to the robot as `safeDstCl`.
  - ```
    ((willClean=TRUE) =>
     (!xx.((xx:(0..7)) =>
      sensorsData(xx)>safeDstCl)))
    ```

- The robot cannot move if anyone gets as close or closer to its front as `safeDstMov`.
  - ```
    ((pos2goDst>0)=>
     (sensorsData(((pos2goAngle/2700)+7)mod 8)
      >safeDstMov &
     sensorsData(pos2goAngle/2700)
      >safeDstMov &
     sensorsData(((pos2goAngle / 2700)+1) mod 8)
      >safeDstMov))
    ```



Verified controller components



Command:

1 – go to a dirty position and clean

3 – go to a parking position and turn off

Verified controller details

- https://hron.fei.tuke.sk/~korecko/sustr22/cleanbotControllerBMethod.zip
  - Specification of the verified controller in B-language
  - The full code of the components

- Tools needed
  - https://www.atelierb.eu/en/
    - B-Method Tool
    - If you would like to try to prove the verified controller by yourself
  - https://hron.fei.tuke.sk/~korecko/FMInGamesExp/resources/BKPICompiler.zip
    - BKPI compiler (Java application)
    - To translate implementations in B-language to JavaScript

Cleaning Robot Case Study

Assignment & Technologies

Development Process Overview

Verified Controller Development

Building Virtual Environment for Validation

Building Virtual Environment

- Web-based virtual reality technologies used
  - A-Frame, https://aframe.io/
  - Networked-Aframe, https://github.com/networked-aframe/networked-aframe

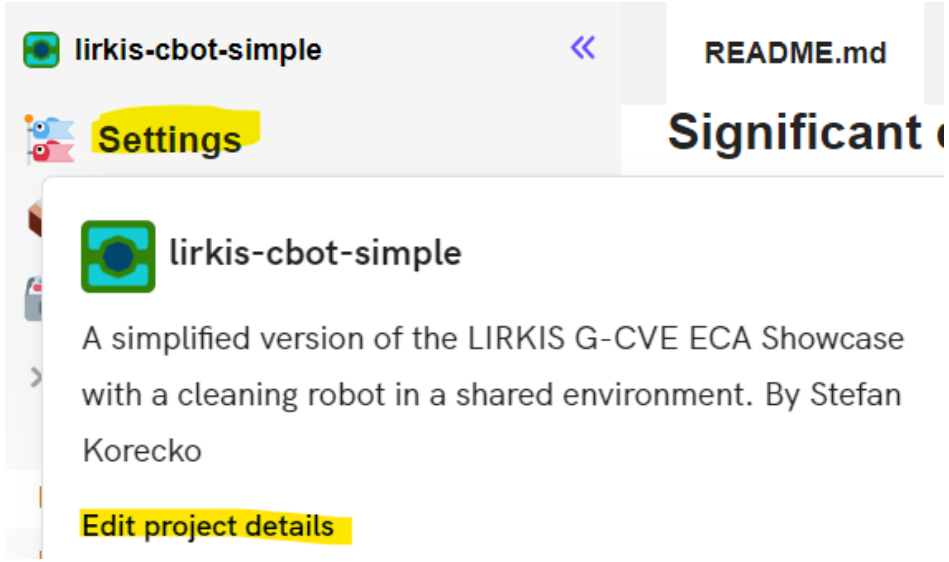
- How to start building the environment
  A. Online, at https://glitch.com/
  B. Offline, using node.js

A. Building Virtual Environment Online

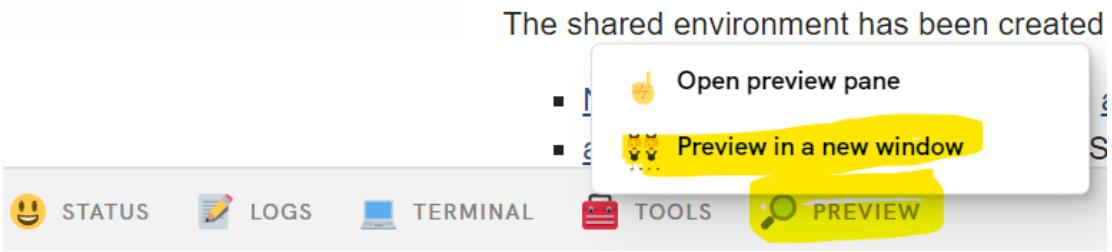
1. If you do not have one, create an account at https://glitch.com/ .
2. Log in at https://glitch.com/
3. Go to https://glitch.com/edit/#!/lirkis-cbot-simple-starter
4. Hit the Remix button in the top right part of the page.

- Next, you can

Rename the project:



Run the project (index.html):



B. Building Virtual Environment Offline

1. If you don't have node.js, download it from https://nodejs.org/en/download/ and install it with standard settings
2. Create a folder for your project
3. Unpack the archive http://hron.fei.tuke.sk/~korecko/sustr22/lirkis-cbot-simple-starter.zip to the created folder
4. Go to the created folder and from the command line run `npm install`

- Next, you can
  1. Run the server with the command `npm start`
  2. Open the page in browser
     - http://localhost:8080/
     - http://localhost:8080/cbot.html

Thanks …

- Thank you for your attention.
- Questions?

Project No. 2020-1-PT01-KA203-078646
Promoting Sustainability as a Fundamental Driver in Software Development Training and Education

Co-funded by the Erasmus+ Programme of the European Union