

Software Engineering

Luís Pedro Coelho

Programming for Scientists

April 7, 2009



University of Pittsburgh

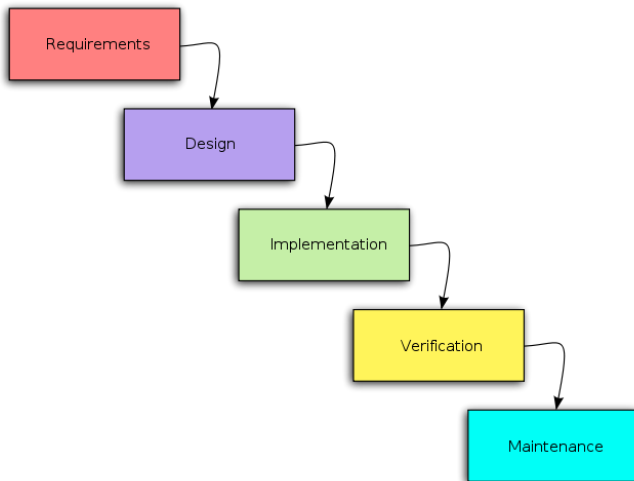
Carnegie Mellon

Most projects that we are tackling do not need software engineering.

Waterfall Model

- 1 Requirements
- 2 Design
- 3 Implementation
- 4 Testing
- 5 Maintenance

Waterfall



Iterated Waterfall

- 1 Requirements
- 2 Design
- 3 Implement
- 4 Testing
- 5 Maintenance
- 6 Goto 1

Requirements for Particles

Major Goals

- Implement particle simulation using *Brownian* dynamics.
- Generate a video containing shot noise.
- Detect particles using global thresholding
- Track using Hungarian algorithm
- Compared inferred std. dev. with underlying std. dev.
- ...

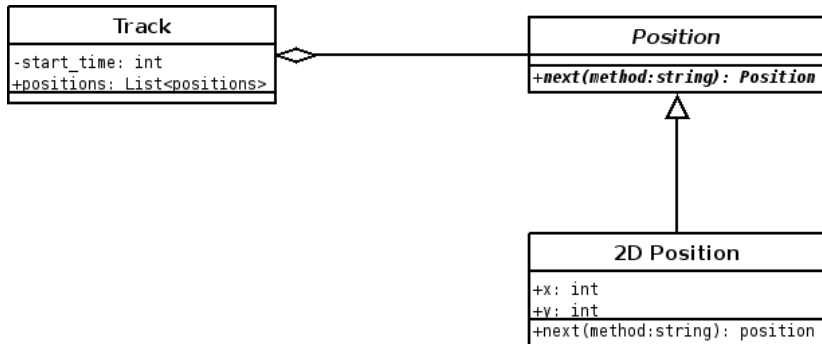
Minor Goals

- Allow the user to set the random seed.
- Allow the user to save the video to a file.
- ...

Design of Particles

- List of tracks
- Tracks
- Video
- Positions

Unified Modeling Language



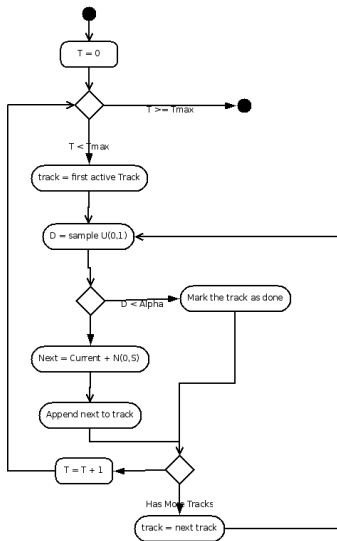
```

class Track:
    def __init__(self) :
        self.positions = None # List<positions>
        self.start_time = None # int
        pass

class Position:
    def __init__(self) :
        pass
    def next (self, method) :
        pass # returns Position

class 2D Position (Position) :
    def __init__(self) :
        self.y = None # int
        self.x = None # int
        pass
    def next (self, method) :
        pass # returns position

```



Design of Particles: Our Version

- List of tracks: simple Python list
- Tracks: $(t_0, [p_0, p_1, \dots])$
- Video: numpy 3-d array
- position: class Position

Use Case 1

Rita is a biologist. She does not care much about algorithms, but wants to know how fast some particles move inside the cell. She has collected some images and applied the Hungarian algorithm for tracking. Before publishing the results, she wants to have validate her implementation. She manually tunes the parameters of the video until it looks like her real particles. She is computer savy, but does not know how to program.

Use Case 2

Wang is a computer scientist. He is interested in testing his new fancy algorithm for particle tracking and comparing it with the traditional Hungarian algorithm. He is comfortable with programming.

Agile Methodologies

- Appeared in the mid-90's.
- Focus on avoiding “Big Design Up Front.”

Properties of Agile Methodologies

- No big design up front.
- Pair programming (two people programming at once).
- Test everything, test it twice.
- Dynamic languages (Python, Ruby, Basic, . . .).
- Release early, release often.
- Avoid over-engineering.
- Organic growth instead of design.
- Constant refactoring

Release Early, Release Often

Short release cycles, even if internal.

Refactoring

- Refactoring means **improving the code** without **changing the program**.
- It often means paying down **technical debt**