Product Planning

Blocks World for Teams

BW4T Context Project

Delft University of Technology

Product Planning

Blocks World for Teams

by

BW4T Context Project

in partial fulfillment of the requirements for the completion of

TI2805: Context Project
of the Bachelor of Computer Science and Computer Engineering

at the Delft University of Technology,

Daniel Swaab dswaab 4237455	Valentine Mairet vmairet 4141784	Xander Zonneveld xzonneveld 1509608	Ruben Starmans rstarmans 4141792
Calvin Wong Loi Sing cwongloising 4076699	Martin Jorn Rogalla mrogalla 4173635	Jan Giesenberg jgiesenberg 4174720	Wendy Bolier wbolier 4133633
Joost Rothweiler jrothweiler 4246551	Joop Aué jaue 4139534	Katia Asmoredjo kasmoredjo 4091760	Sander Liebens sliebens 4207750
Tim van Rossum trvanrossum 4246306	Arun Malhoe amalhoe 4148703	Seu Man To sto 4064976	Nick Feddes nfeddes 4229770
Sille Kamoen skamoen 1534866	Shirley de Wit shirleydewit 4249259	Tom Peeters tompeeters 4176510	

Delft University of Technology

Abstract

Before we can actually start developing, we have to plan ahead; write user stories, schedule releases and create a backlog. The user stories can be used to check whether our implementation truly satisfies the need it had to fulfill.

CONTENTS

1 Introduction

This report describes the product planning. It is built up by first giving a high-level product backlog, where the features that the stakeholders want are listed by their priority along with their relative time estimates. After establishing the items on the high-level backlog, the roadmap will be discussed. The roadmap describes how the product will be built and delivered over time, along with the release goals using a major release schedule.

The product backlog section will contain the initial release plan and the user stories that describe some of the features we would like to include in this release. In order to get a better understanding of the product's users while writing the user stories, some personas were created. The user stories themselves will be about the product's features, defects, technical improvements and know-how acquisitions.

Finally the Definition of Done will be described by stating how many backlog items need to be finished, how frequent we aspire to have releases and how this is integrated with our sprints.

2 Product

2.1. High-level product backlog

User story	Type	Priority	Estimate
As a user I want to be able to understand the BW4Tv3 code base quickly so that I can make changes to the code easily.	Technical work	1	10
As a user I want to be able to manage the capabilities of bots so that I can create bots that have different capabilities.	Feature	2	5
As a user I want to be able to generate one or more MAS files so that I can run various types of simulation runs.	Feature	3	5
As a user I want to be able to create various environment types so that I can build/play/vary the physical realism of the environment.	Feature	4	10
As a user I want to be able to use an e-partner so that I can communicate with other users.	Feature	5	5
As a user I want to be able to be able to control a bot so that I can tell the bot where to go and interact with other players.	Change	6	9
As a user I want the bots to be able to find paths so that the bots can effectively walk around without blocking each other.	Defect	7	4
As a user I want to the bots to have have different channels of communication so that I can regulate communication between the bots.	Change	Low	5
As a user I want to be able to do batchruns so that I can run multiple configurations without having to run them manually.	Feature	Low	3

2.2. Roadmap

- Week 4
  - Restructuring
  - Scenario GUI

  v3.1

- Week 5
  - Bot store

v3.2

- Week 6
  - Environment Store
  - Pathfinding and botcollision

v3.3

- Week 7
  - E-Partner

v3.4

- Week 8
  - Human GUI ease-of-use

v3.5

- Week 9
  v3.6

- Week 10
  v4.0

3

PRODUCT BACKLOG

3.1. USER STORIES OF FEATURES

Story 1.1: Configuration Files
As a user
I want to make configuration files
So that I can easily load configurations

Scenario 1.1.1: Maps
Given the user has launched the GUI
And the user has created the map they wanted
When the user presses the create map button
Then a configuration file should be created which can be used to load and run a simulation

Scenario 1.1.2: Bots
Given the user has launched the GUI
And the user has specified how many bots
When the user presses the create map button
Then a configuration file should be created which can be used to load and run a simulation

Scenario 1.1.3: Human
Given the user has launched the GUI
And the user has specified which bot is human controlled
When the user presses the create map button
Then a configuration file should be created which can be used to load and run a simulation

Story 1.2: Human Player GUI
As a user
I want to control the bot
So that I can tell the bot what to do

Scenario 1.2.1: Control
Given the human player GUI has started
When the user clicks a command
Then the bot executes that command and the GUI shows this as feedback

Scenario 1.2.2: Message
Given the human player GUI has started
When an E-Partner message has been sent
Then the GUI should show this message

Scenario 1.2.3: Visual Feedback
Given the human player GUI has started
When an event has occurred that limits or extends the bots commands
Then the list of possible commands in the GUI should be updated

Story 1.3: E-Partner
As a user
I want to be able to use my E-Partner
So that I can assist bots and use the E-Partner as my assistant

Scenario 1.3.1: Control
Given the user holds the E-Partner
When the user executes an action to move the block representing the E-Partner
Then the block moves in the given direction

Scenario 1.3.2: Communication
Given the user holds the E-Partner
When the user types in a message to send to another E-Partner
And the user clicks to send the message
Then the typed message is sent

Story 1.4: Simulation
As a user
I want to create simulation conditions
So that I can easily create specific scenarios

Scenario 1.4.1: Gripper Handicap
Given the handicap GUI has started before the simulation run
When the user unchecks the ability for a bot to use its gripper
Then when the simulation runs the bot should not be able to use its gripper

Scenario 1.4.2: Colorblind Handicap
Given the handicap GUI has started before the simulation run
When the user unchecks the ability for a bot to see colors
Then when the simulation runs the bot should not be able to differentiate between colors

Scenario 1.4.3: Bot size
Given the handicap GUI has started before the simulation run
When the user changes the size of a bot
Then when the simulation runs the bot should have that size and its restrictions

Scenario 1.4.4: Bot speed
Given the handicap GUI has started before the simulation run
When the user changes the speed of a bot
Then when the simulation runs the bot should have that speed

Scenario 1.4.5: Ability to walk
Given the handicap GUI has started before the simulation run
When the user unchecks the ability for a bot to walk
Then when the simulation runs the bot should not be able to use its gripper

Scenario 1.4.6: Physical aspects
Given the handicap GUI has started before the simulation run
When the user checks the physical aspects of a bot
Then when the simulation runs this bot can only continue its path as long as it does not collide with other bots who have the physical aspects checked as well

3.2. USER STORIES OF TECHNICAL IMPROVEMENTS

Story 3.1: Repast Stepper
As a researcher into multi-agent systems
I want to have a consistent simulation environment
So that I can always reach the same final state if I were using agents with a deterministic algorithm.

Scenario 3.1.1: Equal result
Given the researcher has run a simulation using deterministic agents
And the result was X

When the researches re-runs the simulation
Then the result should be X

Scenario 3.1.2: Equal states
Given the researcher is running a simulation on two seperate systems using deterministic agents
When the researcher pauses the simulation at a specific tick
Then the states on both computers should be equal

3.3. USER STORIES OF KNOW-HOW ACQUISITION

Story 4.1: Repast
To look further into the possibilities of using Repast we have to do extra exploration work. Repast is used throughout the BW4T codebase, but is essential for the simulation to work. Right now Repast is mostly used to keep track of objects and their locations in the continuous space. Furthermore, the "moveTo" method of the continuous space is indirectly used to move the robots (they are drawn by using the location on the grid), but the pathfinding isn't done by Repast. In fact, Repast does not have a pathfinder. In Repast agents are moved by using either the "moveTo", "moveByDisplacement" or "moveByVector" methods. In a nutshell, Repast is currently used as a collection of all objects in the simulation space, and to map objects to a certain point in the simulation space. The fact that Repast isn't needed that much at the moment, doesn't mean that we should get rid of it. It offers some features such as the RandomCartesianAdder class which allows for adding objects at a random location in a space, and which could therefore be used to generate random maps. In addition it has support for getting specific objects or agents from neighbourhood grid spaces. Therefore we should further explore the possibilities of Repast in improving our system.

Story 4.2: Existing code analysis
The code we have received to work with was not written by us and therefore requires our time to analyse what is already in the code and what should be changed or left out. We know that the current version of the source code is unstructured, difficult to work with and uses an enormous hashmap. Before we can refactor the current code we should first look into the code architecture and understand it.

3.4. INITIAL RELEASE PLAN

The overall project has a lot of features that the stakeholders wish to see. The most important milestones of these features are:

- Restructuring
- Scenario GUI
- Bot Store
- Environment Store
- Pathfinding and Bot Collision
- E-Partner
- Human GUI

Per release there is a certain minimum of releasable features that has to be met in order to meet the expectations of the customer. Therefore these features have to be finished with the release they correspond to. For each release these features are as follows:

- Sprint 1
  - Restructuring
  - Scenario GUI
- Sprint 2
  - Bot Store
- Sprint 3
  - Environment Store

  - Pathfinding and Bot Collision
- Sprint 4
  - E-Partner
- Sprint 5
  - Human GUI

4 Definition of Done

Now that we have acquired many different tasks, we have sorted them in the categories mentioned below. We will be using the following definition of done to decide when a task has been completed:

1. Tasks
   (a) User story clarity achieved
   (b) All tasks are handled in Trello
   (c) All tasks are on status “Done”
   (d) OK from Product Owner
2. Meetings
   (a) Short demo prepared and verified before SE aspects meeting
3. Sprints
   (a) Sprint plans are done
   (b) Code integration is done in Jenkins
   (c) Jenkins built version and all tests pass
4. Code
   (a) Code is working - no errors
   (b) Code is well commented
   (c) Code is well documented
   (d) Unit tests pass
   (e) Checkstyle is approved
5. Testing
   (a) Unit tests are written
   (b) Checkstyle xml is written
   (c) Functional testing
   (d) Regression testing
   (e) Performance testing
   (f) User tests defined
6. Review
   (a) Peer review from each party
   (b) Software quality review
   (c) Code review in pairs
   (d) Good overall test coverage (75%)

GLOSSARY

deterministic algorithm is an algorithm which, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states. . 7

E-Partner a small electronic device which can assist a robot if it's possessed by the robot. . 5–9

Pathfinding the act of computing a path, often with a predefined algorithm such as Dijkstra's algorithm, along which the robot should travel. . 5, 8, 9