Product Planning
Blocks World for Teams

BW4T Context Project

Delft University of Technology

TU Delft Delft University of Technology

Challenge the future

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	

An electronic version of this document is available at https://github.com/MartinRogalla/BW4T/.

TU Delft 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 frequent we aspire to have releases and what needs to be finished at each release.

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 use an e-partner so that I can communicate with other users.	Feature	3	5
As a user I want to be able to generate one or more configuration files so that I can run various types of simulation runs.	Feature	4	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	5	5
As a user I want to be able to create various environment types so that I can build/play with/vary the physical realism of the environment.	Feature	6	10
As a user I want to be able to control a bot so that I can tell the bot where to go and interact with other players.	Change	7	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	8	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

2.2. ROADMAP

The roadmap starts at week 4, the start of the implementation. Below is our roadmap.

- Week 4
  - Launcher
  - Core - Client - Server structure
  - Scenario GUI design.
  - Implement Scenario GUI design.

  - Bot Store GUI design.
  - Human player GUI design.
  - Implement Bot Store GUI design.
  - Beginning with Battery Functionality.

v3.1

- Week 5
  - Documentation codeBase complete on package level.
  - Wouter must know what the new architecture is.
  - First integration with other teams.
  - Make sure that the stakeholders get why we do what we do (eg. with repast).
  - Implement Scenario GUI design with feedback.
  - Finish Battery Functionality.
  - Interface for handicaps.
  - Implement handicaps.
  - Beginning with design and implementation E-Partner.

v3.2

- Week 6
  - Exception handling.
  - Sonar issues.
  - More specific documention codebase (core classes).
  - Visualization package for Client en Server mergen.
  - Implement E-Partner functionality.
  - Completely finish E-Partner.
  - Completely finish Bot Store.
  - Environment Store GUI design.
  - Complete Scenario GUI.
  - Implement reading and loading config file.
  - Implement batch runs.
  - Integrate Scenario Editor GUI and Bot Store GUI.

v3.3

- Week 7
  - Good graphical overview.
  - Documentation codebase (complete class doc).
  - Make an overview of the system with CodeCity.
  - Analyse tool wich uses the logger.
  - Implement Environment Store design.
  - Environment store: Manual map maker.
  - Environment store: Random map maker.
  - Completely finish batch runs.
  - Implement Human player GUI design.

v3.4

- Week 8
  - System works the same as before refactoring, functionalities are the same.
  - Fully integration tests.
  - Test coverage of 50% over the whole system.
  - Completely finish Human player GUI.
  - Completely finish Environment Store.
  - Put the Environment Store GUI together with the other GUI's.
  - Physical realism: Pathfinding/collision avoidance.

v3.5

- Week 9
  - Final bug fixes.

v3.6

- Week 10
  - Presentation final product.

v4.0

3

PRODUCT BACKLOG

3.1. USER STORIES OF FEATURES

Story 1.1: Configuration Files (Scenario GUI)

As a user
I want to have a scenario GUI
So that I can easily make custom configurations that are read by GOAL to create maps

Scenario 1.1.1: Configuration changing
Given the user has started the program
When the user has launched the GUI
Then the user can change the settings from a standard map to create a map with an entirely new set of configurations

Scenario 1.1.2: Room count
Given the user has changed the room count
When the user presses the create map button
Then the configuration file created contains that amount of rooms
And the map shows that amount of rooms when loaded as environment

Scenario 1.1.3: Bots
Given the user has launched the scenario GUI
And the user has specified how many bots
When the user presses the create map button
Then the system creates a configuration file containing the settings

Scenario 1.1.4: Human Players
Given the user has launched the scenario GUI
And the user has specified which bot is human controlled
When the user presses the create map button
Then the system creates a configuration file containing the settings

Scenario 1.1.5: Maps
Given the user has launched the scenario GUI
And the user has created the map they wanted
When the user presses the create map button
Then the system creates a configuration file corresponding with the map

Story 1.2: Human Player GUI

As a user
I want to control the bot via the human player GUI
So that I can tell the bot what to do

Scenario 1.2.1: GUI starting
Given a map with at least one human player has been created

When the map is used in a GOAL multi agent system
Then there should be a human player GUI launched for every human player given in the map

Scenario 1.2.2: Control
Given the human player GUI has started
When the user clicks a command given in the list of possible commands
Then the bot executes that command and the GUI shows this as feedback

Scenario 1.2.3: Message sending
Given the human player GUI has started
When the user makes a message and sends it
Then the GUI should show this message as being sent

Scenario 1.2.4: Message receiving
Given the human player GUI has started
When the user receives a message made by another human player GUI
Then the GUI should show this message as being received

Scenario 1.2.5: 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: Starting the E-Partner
Given the GOAL multi agent system is launched
And there is an E-Partner in the map
When a bot picks up the E-Partner
Then the E-Partner should be started and all its features should be unlocked for the bot that has picked it up

Scenario 1.3.2: 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 1.3.3: Communication (sending)
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

Scenario 1.3.4: Communication (receiving)
Given the user holds the E-Partner
When another bot or E-Partner sends a mesage to this E-Partner
Then the message is shown to the user of the E-Partner

Scenario 1.3.5: GPS functionality
Given the user holds the E-Partner
When the GPS functionality of the E-Partner is used
Then the robot knows where it is on the map

3.2. USER STORIES OF TECHNICAL IMPROVEMENTS

Story 2.1: 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 deterministic agents.

Scenario 2.1.1: 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 2.1.2: 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

Story 2.2: Simulation (Handicaps)
As a multi-agent systems researcher
I want to create simulation conditions
So that I can easily create specific scenarios

Scenario 2.2.1: Bot store GUI
Given the user has launched the scenario GUI
When the user selects a bot from the list of bots in the scenario GUI
Then the bot store GUI should be launched containing the settings for the selected bot

Scenario 2.2.2: Gripper Handicap
Given the bot store GUI has started before the simulation run
When the user unchecks the ability for a bot to use its gripper
Then during the simulation the bot should not be able to use its gripper

Scenario 2.2.3: Colorblind Handicap
Given the bot store GUI has started before the simulation run
When the user unchecks the ability for a bot to see colors
Then during the simulation the bot should not be able to differentiate between colors

Scenario 2.2.4: Bot size
Given the bot store GUI has started before the simulation run
When the user changes the size of a bot
Then during the simulation the bot should have that size and its restrictions

Scenario 2.2.5: Bot speed
Given the bot store GUI has started before the simulation run
When the user changes the speed of a bot
Then during the simulation the bot should have that speed

Scenario 2.2.6: Physical aspects
Given the bot store 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

3.3. USER STORIES OF KNOW-HOW ACQUISITION

Story 3.1: 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 3.2: 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

- Week 4
  - User can click on buttons and fill in text fields in the Scenario Editor.
  - User can use all the functionalities of the Bot Store GUI.
  - User can save the data from the Bot Store UI.

  v3.1
- Week 5
  - User can add, modify and delete bots and e-partners.
  - User can use all the functionalities of the Bot Store GUI.
  - User can save the data from the Bot Store UI.

  v3.2
- Week 6
  - User can add, modify and delete bots and e-partners.
  - User can use both the Scenario UI and the Bot Store UI together.
  - User can make batch runs.

  v3.3
- Week 7
  - User can see the Environment Store GUI and already use some of the functionalities.
  - User can see the Human Player GUI and already use some of the functionalities.

  v3.4
- Week 8
  - User can control a bot using the Human Player GUI.
  - User can use all the functionalities of the Environment Store GUI.
  - Bots can find alternative paths if they block each other.
  - User can access all the different UI's.

  v3.5
- Week 9
  - User can run simulations combining all the UI's.
  - User can see the robots moving around in the environment.
  - System works the same as before refactoring.

  v3.6

- Week 10
  - Presentation final product.

v4.0

4

Definition of Done

Now that we have acquired many different tasks, we have sorted them in the categories mentioned below. The duration of our sprints are one week. After every sprint there will be a release. 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%) over the new code

GLOSSARY

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

Pathfinding the act of computing a path, often with a predefined algorithm such as Dijkstra's algorithm, along which the robot should travel. . 6