Using BDI-extended NetLogo Agents in Undergraduate CS Research and Teaching

Jonathan Wiens
Computer Science Dept.
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
j.wiens@stud.hwr-berlin.de

Dagmar Monett*
Computer Science Dept.
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
Dagmar.Monett-Diaz@hwr-berlin.de

***Abstract*—This paper introduces both the subject and the research results of an undergraduate student project. The project focuses on an agent architecture that implements the beliefs-desires-intentions (BDI) model. It proposes a new way to extend a BDI library in NetLogo based on a case study from Computational Economics. Furthermore, this work presents how such an undergraduate research supports Artificial Intelligence teaching activities at the Berlin School of Economics and Law.**

***Keywords*—Multi-agent systems, BDI agents, NetLogo, undergraduate research, teaching.**

I. Introduction

In dynamic environments like economic models, telecommunication networks or biochemical organisms, simulation processes often rely on multi-agent systems (MAS). MAS play a major part both in Artificial Intelligence (AI) curriculum and research. They are a basis for many commercial models simulating complex non-deterministic behavior. Just as developing simulations of complex environments with MAS is an inherent difficult task, teaching by undergraduates and conducting programming MAS is not an easy one, due to a necessary high time exposure for MAS learning, design and implementation.

The rest of the paper continues as follows: Section II presents both a theoretical background of the beliefs-desires-intentions (BDI) model and NetLogo, a multi-agent programmable modeling environment. In Section III, a case study from the area of Agent-based Computational Economics (ACE) is introduced. Fractional reserve banking supports the need to extend an existing BDI library for NetLogo, which is the topic of an undergraduate research project at the Berlin School of Economics and Law (BSEL). It is presented in Section IV. Finally, Section VI gives insights concerning our experiences in undergraduate Computer Science (CS) education, when teaching these subjects in an AI course at the BSEL.

II. Theoretical background

This section introduces a practical reasoning model for agents with future-oriented intentions, the BDI model in NetLogo. NetLogo [1] is a multi-agent programmable modeling environment that aims at providing standards for high power users at low learning curves. Thus, it serves as an excellent teaching tool to discover basic concepts of agent programming with a hands-on approach.

To date, a third-party library providing functions for the implementation of BDI agents written in native NetLogo code has been published [2]. This library, however, only includes belief revision procedures and intention handling and execution, thus missing other elemental features that the BDI approach encompasses. Subsequently, the main topic of the undergraduate student project was to enhance the existing library in order to acquaint students with a more sophisticated form of the BDI model in NetLogo. Furthermore, the improved library can be used not only by students but also by anyone interested in building BDI agents in NetLogo.

The major goals of the undergraduate student research project are:

- to propose a simplified model for fractional reserve banking,
- to integrate BDI concepts,
- to design a MAS in NetLogo for the new model,
- to analyze which BDI components cannot be modeled using the available BDI library,
- to propose extensions to that BDI library for covering the discovered needs,
- to design, implement, test and evaluate those changes using NetLogo, and
- to support the AI teaching of these contents both theoretical and practically.

All these goals were successfully achieved and are topic of the sections that follow.

A. BDI

In 1987, the philosopher Michael Bratman introduced a model for human practical reasoning as a way to explain future-directed intentions [3] which led to the BDI model in AI. First, a logic was formally instituted in 1991 [4] and later, a practical model was implemented as an approach to build highly dynamic applications, like the OASIS air-traffic management system, the Optimal Aircraft Sequencing using Intelligent Scheduling [5].

* Contact author.

The BDI approach focuses on the reasoning of resource-bounded agents, while resource boundaries include limited computational power of both the agent and the dynamic environment. This implies that (i) while the environment changes, the agent cannot take infinite time to plan its actions but needs to be flexible in re-planning and (ii) the agent may not reason open-ended because of world changes that lead to new dangers or opportunities.

At the center of such BDI agents lies the notion of mental attitudes, namely beliefs, desires and intentions. Beliefs are the agents representation of assumptions about the world, the environment. Beliefs are not facts about the world, since beliefs are subject to error of reception. Thus, beliefs can change and can be combined to infer new beliefs. Desires represent the goals of the agent and thereby ultimately define an agent's purpose. An agent can have different goals, competing for each other. One difficulty in BDI is how to distinguish which goals are most *beneficial* to the agent. The substance of desires are intentions, which describe how to reach the deliberated goals, like directions in a recipe. While it can be assumed that for most desires there are several intentions, a selector is needed to determine which intention is appropriate at a given time. Intentions are, in other words, the actions necessary to fulfill the agent's goals. A sequence of intentions is called a plan.

B. NetLogo

If BDI is an architecture describing how to build a specific type of agents, then NetLogo is the software developer tool kit providing a framework to design the agents and their environment. NetLogo is a multi-agent programming language with an integrated development environment (IDE) for simulating complex phenomena for both research and educational purposes. It is used across a wide range of disciplines and university levels and is particularly well suited for multi-agent systems that evolve over time.

Compliant with it's motto "low threshold, no ceiling", NetLogo allows the modeling of hundreds and thousands of autonomous agents, all operating concurrently. The graphical user interface allows the developing of quick simulations of two and three dimensional spatial behavior of agents, making the process of designing MAS visual and therefore easier to understand and troubleshoot, but also entertaining and appealing [1].

To date, there are several languages for programming BDI agents that are highly sophisticated and offer different components and approaches to implement multi-BDI agent systems [6]. These environments, though powerful and advanced, have steep learning curves and are unsuitable for educational purposes that involve a hands-on, short-term curriculum. Because NetLogo is an excellent tool for teaching, for researching and for designing MAS, the notion to create BDI agents with NetLogo comes natural.

C. BDI Library for NetLogo

The library for realizing BDI agents in NetLogo was written by Sakellariou [2] and its current version is NetLogo 4 compatible. It consists of 23 *commands* and *reporters*[1] and is split into three logical tiers: *belief management*, *intention management* and *utilities*. Utilities support the belief and intention management with, for example, timeout interruptions.

1) Belief Management: A belief consists of two elements, stored in a list: the *type* and the *content*. The type of the belief describes what kind of belief it is or what class it belongs to. For example, a type of a belief could be any string like `"location"` or `"capital"`.

The content declares the actual value of the belief type and can be of any NetLogo structure.[2] Notice that there can be competing beliefs with the same type but different content. This makes sense when a belief can hold multiple attributes. An agent could believe, for example, that the coffee is hot (type `"coffee"`, content `"hot"`) and strong (type `"coffee"`, content `"strong"`) at the same time.

The following example shows three sets of beliefs, as part of one list:

```
[["location" [3 4]] ["coffee" "hot"]
 ["coffee" "strong"]]
```

2) Intention Management: Like a belief, an intention consists of two elements: the *name*, which maps to a NetLogo command, and a *done-condition*, which maps to a NetLogo reporter. Intentions are stored in a stack and are popped out when to be executed. If the done-condition is satisfied, the intention is removed and the next intention is popped out consecutively.

An example intention stack in NetLogo is the following:

```
[["task [do-nothing]" "true"]
 ["task [move-to [0 3]]"
  "task [at-location [0 3]]"]
 ["task collect-wood" "task wood-loaded"]]
```

The name and done-condition of an intention are stored in *tasks* – a task is a special NetLogo structure that is a placeholder for runnable code, in other programming languages also known as lambda, closures or first-class functions.

III. A Case Study

Having covered so far the theoretical background of BDI, NetLogo and the available NetLogo BDI library, the next step is to use the library by designing a sufficient complex model in order to exhaust the library's functionalities and, consequently, to demonstrate in what ways it can be extended. The Agent-based Computational Economics (ACE) interdisciplinary research field promises scenarios that can be as complex as human action itself and therefore suits the purpose of building a complex model in NetLogo.

[1]In NetLogo, commands describe the actions an agent should carry out resulting in some effect, while reporters include instructions to compute a value, which is then reported by the agent.

[2]Like booleans, strings, integers, doubles, lists, etc.

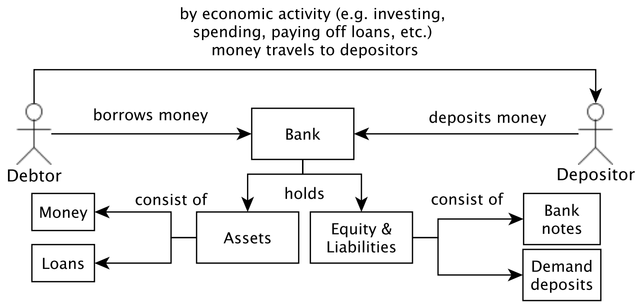

Fig. 1. Fractional reserve banking simplified.

A. Agent-based Computational Economics

Agent-based computational economics is “the computational study of dynamic economic systems modeled as virtual worlds of interacting agents” [7], while agents can represent individuals (e.g. people) or abstract entities (e.g. a market interest rate). Like with other MAS, the modeler provides the agents with an initial configuration and lets the computational world evolve over time, as its constituent agents interact with each other. Initial configuration, depending on the subject and desired complexity of the model, includes any variable that is specified to reach the desired goal. Hence, the scientific goal is to “[...] test theoretical findings against real-world data in ways that permit empirically supported theories to cumulate over time” [8].

ACE has been applied to several research areas, like asset pricing [9], transaction costs [10], and macroeconomics [11], to name a few. The economic subject of the NetLogo model we present is this paper is fractional reserve banking (FRB), which offers an arbitrary amount of complexity depending on the modeler’s choice.

B. Fractional Reserve Banking

In economics, the chief activity of a bank is, as a financial intermediary, to receive deposits from savers at a given deposit interest rate and to channel these deposits by lending them at a higher interest rate to credit-worthy or productive borrowers, thus making a profit. FRB is the process when banks keep only a fraction of the received deposits, called the reserves, and lend the remaining to borrowers. Subsequently, by spending, investing or doing other economic activity, part of the money lent is deposited within the same or another bank, allowing further lending (see Figure 1).

As a result, the bank holds more liabilities than assets at a growing rate. This process increases the money supply over time, leading to inflation and currency debasement. The degree of the effects is mainly influenced by the reserve ratio, which is called the reserves-to-warehouse-receipts quotient [12]:

$$\text{Reserve ratio} = \frac{\text{Reserves}}{\text{Warehouse receipts}}$$

Warehouse receipts are the outstanding bank notes the bank is required to redeem on demand to the depositor, usually stored in a checking account. The depositor is able to withdraw on his account anytime and anywhere checks are accepted as payment. However, if the liabilities exceed the assets, then the depositor will not be able to withdraw his entire deposit, since the bank only holds a fraction of it, hence the name fractional reserve banking.

TABLE I. EXAMPLE BALANCE SHEET OF FRACTIONAL RESERVE BANKING.

Bank			
Assets		Equity & Liabilities	
Gold	\$50,000	Warehouse	
Loans	\$80,000	receipts for gold	\$130,000
Total Assets	\$130,000	Total Liabilities	\$130,000

a) An example for FRB: In the absence of a central bank, and for the sake of simplicity, a bank has \$50,000 of gold[3] deposited and issues \$80,000 in warehouse receipts, it lends them out and expects to be repaid the \$80,000 plus interest (see Table I).

The bank lends \$80,000 of warehouse receipts it doesn’t own. The general money supply is thus increased by the amount of the credit (i.e. \$80,000), the fraction being $\frac{5}{13}$ (from $\frac{\$50,000}{\$130,000}$). Thus, FRB generates an increase in the money supply by issuing warehouse receipts for money that did not exist previously. Money in circulation has increased by the amount of warehouse receipts issued beyond the supply of gold in the bank. The lower the fraction of the reserve, the greater the amount of new money issued and, therefore, the greater the decrease of the purchasing power of the dollar. This is why FRB has a number of negative effects on the economy, in terms of interest rates, inflation and the purchasing power of the dollar in form of currency debasement. The fact that banks are never able to redeem the entire deposits of all depositors has three implications:

- the bank is at all times bankrupt, since it cannot redeem all the deposits it owes to the depositors, hence
- the depositors statistically never demand to redeem all their deposits at once, unless
- they lose confidence, or trust, in the bank and, therefore, want to redeem all their deposits, resulting in a *bank run*.

A bank run is the phenomena when a large number of depositors withdraw their deposits due to loss of confidence in the bank or when they speculate that the bank might have become insolvent. As the bank run progresses, it develops momentum in a positive-feedback loop as more customers withdraw their deposits, thus prompting further withdrawals from other customers. This ultimately results in the bank’s insolvency, as it fails to provide the money demanded.

IV. NetLogo Model for FRB

The simplified model for FRB derived in a complex enough model for BDI in NetLogo. The purpose of the FRB model is to depict the implications fractional reserve banking has over time. Its current stage is kept to a simple world design without inter-bank activities, a central bank and government regulation.

[3]Whether gold or government paper does not matter here.

As a consequence, this model reflects by no means a real-world example but is merely an extract of modern banking to illustrate, on a low-level, what effects fractional reserve banking has on both interest rates and currency debasement.

Since MAS evolve over time, the FRB model is put into a time frame that is measured in *ticks*, discrete steps that are used in most NetLogo models. Figure 2 shows the model with the concurrent activities that are considered within a tick cycle. At the heart of the model are three parties: the debtors, the depositors and the bank. As described in Section III-B, the bank plays the role of the intermediary between the depositor and the debtor, handling the deposits, loans and their corresponding interest rates. Each party deliberates once per tick. The deliberation process determines, according to the agent's personal value scale (PVS) what it will do. At the end of each cycle there is one possible action per agent.

A. Depositor

The depositor's motivation is to accumulate as much capital as possible without running the risk of the bank's insolvency, which would cause him immediate loss of capital. Therefore, there are four different general actions a depositor can accomplish[4] (see Figure 3).

Since the actions of the depositor are determined by his PVS, the PVS should be designed carefully. It is the result of the agent's algorithm and embodies what would be most reasonable to undertake. It would be unreasonable, for example, for a depositor to deposit 70% of his on-hand cash into a bank he has 10% of trust in. The goal is to project the depositor's most reasonable behavior using an algorithm that takes account all influencing variables, like his time preference p, his current capital C, the deposit interest rate β and the trust in the bank t, in a feasible way. In order to design an algorithm that captures all of these parameters in a balanced relationship, an iterative process of implementing, testing and adapting is necessary. The following pseudo-code shows the result of this process:

```
**if** $0.5 \leq t \leq 1$ **then**
    $D = C \cdot (p + \beta)$
**else if** $0.3 \leq t < 0.5$ **then**
    $W = S \cdot (1 - p)$
**else**
    $W = S$
```

where D is the amount to deposit, W is the amount to withdraw, and S is the current deposits or savings in the bank.

The algorithm relies heavily on the agent's trust into the bank he's engaged with. If it falls to a certain amount, he will refuse to deposit but rather withdraw, even to the point of withdrawing the entire deposit, if the confidence in the bank is lost. Both the time preference and interest rate determine what amount of capital should be deposited or deposits withdrawn. The time preference is an attribute that cannot be deduced by computational methods, since it has a non-deterministic character. Therefore, the time preference is generated through a randomized procedure for each agent individually.

[4]The four actions being to retrieve the entire deposit, retrieving a specific amount that is lower than the total of the deposit, depositing a certain amount, or doing nothing.

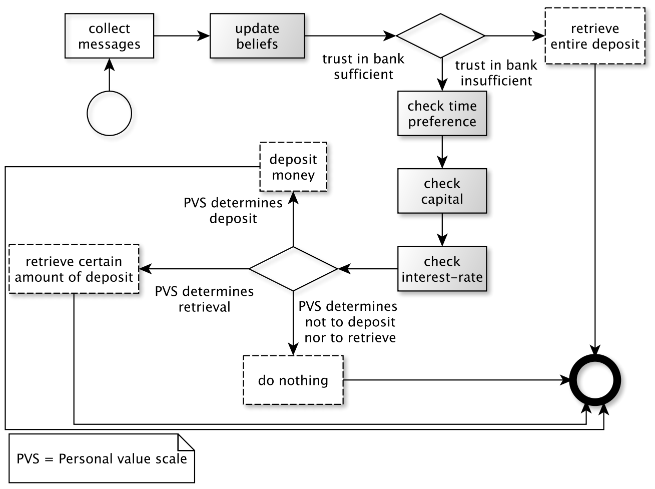

Fig. 3. Deliberation process for the depositor, including evaluation of beliefs (grey boxes) resulting in four general intentions (boxes in dashed lines).

B. Debtor

The debtor's deliberation process is similar to the depositor's. The debtor's four possible actions are to deleverage by a certain amount, to lend a specific or the maximum amount of money, or to do nothing. They were designed and implemented analogously to the depositor (for more information, refer to the student research project documentation [13]).

C. Bank

The bank's main procedures are to accept deposits, to issue withdrawals, to lend, to deleverage and, corresponding to these, to compute profitable deposit and loan interest rates, as well as to compute the reserves, liabilities and assets. The goal of the bank is to make profit through interest rates by maintaining the loan interest rates greater than the deposit interest rate. The figure of each rate is determined by the economic law of supply and demand. For example, if there is more demand than supply in the deposit business, then the bank will decrease its interest rate for deposits. If the demand diminishes, then the interest rate goes up for giving potential customers incentives to deposit.

V. Extention of the BDI Library

The case study in ACE resulted in the proposition for plans and redefining intentions. The original implementation of the BDI library relies solely on intentions for an agent's actions. Because the complexity of agents and the implementation and maintenance of simple intentions correlates, the modeler needs to create a new intention for each action and has no possibility to cluster intentions. Therefore, the primal reason to incorporate plans into the library is to group a sequence of actions, or intentions, and other plans. Like a knowledge base, a plan defines what needs to be done in order for the goal to be fulfilled.

The second major upgrade was the redefinition of intentions. The original implementation of intentions did not allow calls by reference. This has a strong influence on the agent's

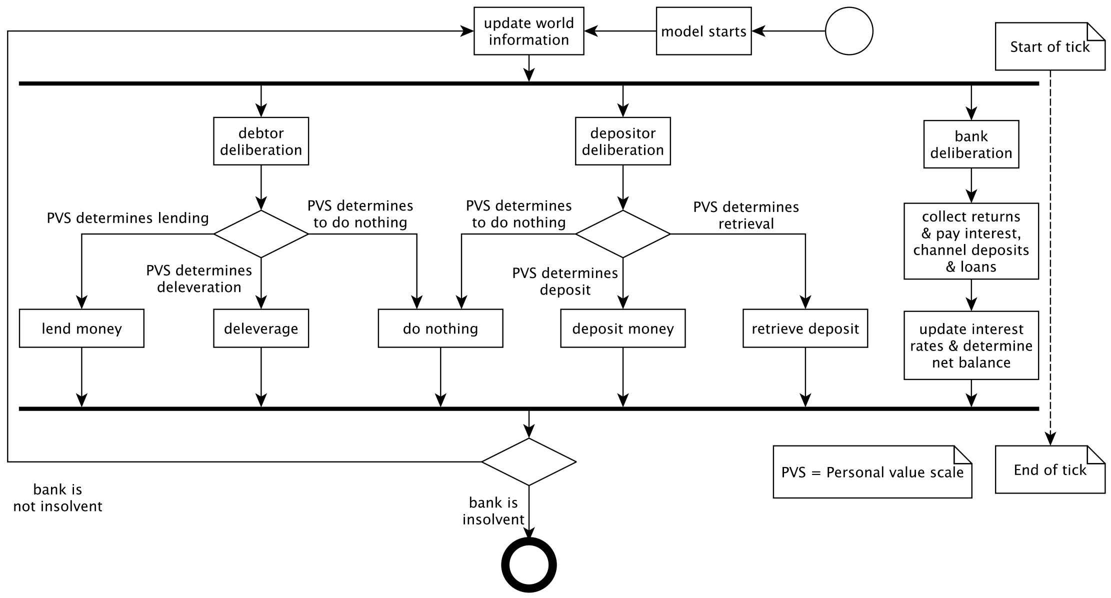

Fig. 2. Fractional reserve banking model overview.

capabilities, since it cannot act on its beliefs in an efficient manner.

An example of a plan as described for the extension can be written as follows:

```
create-plan (list boil-water
add-coffee-powder
stir-beverage drink-coffee)
task [coffee-finished?]
"make coffee"
```

The items `boil-water`, `add-coffee-powder`, `stir-beverage`, and `drink-coffee` refer to intentions that are executed consecutively in order to achieve the plan. The task `coffee-finished?` is the reporter to verify whether a plan has been completed. The field `"make coffee"` is an optional info field.

With this approach, the programmer can combine multiple intentions that achieve one purpose (in this example drinking coffee). Without the extension for plans, the programmer would have put the intentions on a stack manually, making the management of intentions with one purpose difficult to manage. The structure for plans allows dynamic adding, as well as deleting single or multiple intentions at a time.

This is a significant feature for developing more complex projects, for example MAS student projects. In the following sections, we present our experiences in undergraduate CS education when applying the concepts and the BDI extension introduced so far.

VI. Experiences in CS education

The Artificial Intelligence course at the Berlin School of Economics and Law is part of the curricula in the fifth semester of the Bachelor of Computer Science, carrier accredited and recently re-accredited (for 7 more years) by the AQAS e.V.[5] It has two modules: the first course module being *Autonomous agents and multi-agent systems* (33 hours à 45 minutes) and the second course module being *Knowledge-based systems* (44 hours à 45 minutes). The AI course is an optional course that offers 7 ECTS-credits,[6] and that has a final exam as the primary evaluation form, although students can opt for some credits upon team working in a course project. In such a case, concrete assignments that improve both their written and oral skills are required.

The content is taught in lectures following both a teacher and a student-centered approach. They are organized in six weeklong sections of four hours each. The main subjects that are taught in the first module cover much of those in Wooldridge's book [14] on agents and multi-agent systems.[7] Special attention is given to the BDI model, as well as to both the development and the simulation of practical reasoning agents.

A. The Group

Since the AI course is an optional course, it enrolls only a part of all fifth semester CS students at the BSEL. Most AI classes do not exceed 20 students. Teaching small-sized groups has had positive implications when lecturing, selecting and administering exercises, course projects, as well as on didactic methods to be used in the classroom.

B. NetLogo in the Classroom

NetLogo has already been used as an educational tool in numerous university courses worldwide [15]. We have

[5]German agency for the accreditation of study programs.

[6]European Credit Transfer and Accumulation System. In Germany, one credit point is equivalent to 30 hours of study.

[7]Teaching resources including lecture slides and syllabus descriptions that accompany Wooldridge's book can be found at http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/imas/resources.html

also used NetLogo in our AI course since 2006. BSEL's CS students learn how to simulate multi-agent scenarios by programming in NetLogo for different undergraduate course projects [16]. Typical multi-agent scenarios are those of rescue agents simulations and predator-prey simulations, for instance. Extensions to NetLogo have been topic of several student research projects at the BSEL as well.

C. The Sandwich Model

For supporting a wide spectrum of class activities and learning opportunities in the first module of the AI course, topics were taught by three different persons in Summer 2012: a faculty professor, a technical assistant, and an undergraduate student in the role of a teaching assistant. In what follows, we refer only to the part taught by the undergraduate student. He is also the developer[8] of the BDI extension already introduced in this article. A differentiated evaluation schema was considered, since the student was also enrolled in the AI course. For doing this, not answering some final exam questions was offered to gain credits against teaching as part of the teaching staff.

Teaching involving the undergraduate student was devoted to the BDI theory, to BDI agents, and to the use of NetLogo to simulate them. All topics were scheduled and discussed with the faculty professor prior to the lectures. They were taught at two lectures following a sandwich model, i.e., by combining passive and active learning units. In his first talk, the student introduced NetLogo as the supporting tool for the course projects to be developed by the rest of the students. He explained NetLogo's main features and where to find what kind of resources. Such a general introduction took about 10 minutes. He then started a practical section of half an hour of duration, where students modeled from scratch, step by a step, and by following his instructions, a predator-prey scenario: almost all of Netlogo's elements and options were introduced and demonstrated based on the Wolf Sheep Predation model from Wilensky [17]. In his second talk, the student followed the sandwich schedule presented in Figure 4 for introducing BDI-related concepts. The duration time in minutes for each activity is given at the left hand side in the figure.

A sandwich structure includes breathing-in and breathing-out phases, which successfully combine theoretical input and practical assignments, as described in [18]. The video that was shown in the beginning opened with a group discussion to motivate both BDI termini and BDI agents. A passive learning section followed, in which beliefs, desires, intentions, and agent commitments towards goals were introduced, as well as a representation of such terms by using Rao and Georgeff's logic [4]. After that, the students had the opportunity to practice the new acquired knowledge by solving exercises specially prepared by their teaching classmate. Solutions were discussed and presented in plenum and were moderated by the teaching student. Then, BDI architectures were introduced briefly, followed by a concrete, practical example in NetLogo using the BDI model. The Wolf Sheep Predation model shown in the first talk a week before, was now being implemented according to the BDI paradigm. The sandwich schedule ended with some advise for implementing BDI in the course projects

[8]At the time of writing this paper, Jonathan Wiens is an undergraduate student attending the fifth semester in the CS career at the BSEL.

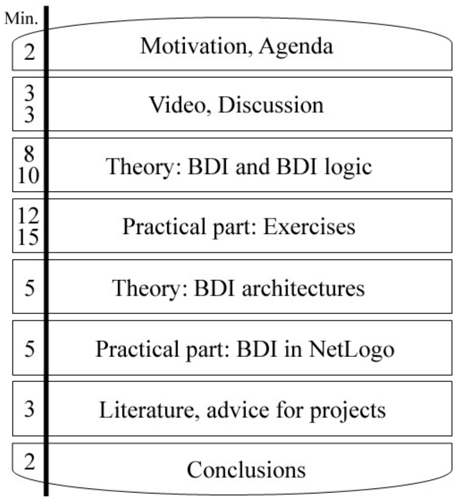
Min.
2 Motivation, Agenda
3 3 Video, Discussion
8 10 Theory: BDI and BDI logic
12 15 Practical part: Exercises
5 Theory: BDI architectures
5 Practical part: BDI in NetLogo
3 Literature, advice for projects
2 Conclusions

Fig. 4. Schedule of a student's talk as a sandwich structure.

(both with and without using the new extension to the existent BDI library) before the conclusions of the talk were addressed.

VII. Evaluation

The extensions to the BDI library that are presented in Section V were analyzed, designed, implemented, tested and evaluated successfully. Further details are described in the student research project documentation as well and are not addressed here since it would go beyond the scope of this paper (for more information we refer to the original research paper [13]).

Since the project has completed and the results presented, the new library has been part of AI teaching in the classroom. Students can now learn not only the theory concerning BDI but also have the possibility to design and implement MAS in NetLogo, especially when using BDI practically. After following the instructions given by their classmate step by step, they were able to incorporate the BDI approach into their course projects, while gaining more diverse programming skills in NetLogo. The support of the teaching student was highly welcomed in this respect.

Moreover, the students were given at the beginning of the course the possibility to choose either to complete a final exam to evaluate the module *Autonomous agents and multi-agent systems*, or to work on a course project using NetLogo and BDI. After attending the teaching student's tutorials, almost all the groups chose to work in projects, which were all qualitatively high and fully completed by the end of the term.

Last but not least, the extended library has been effectively used in other simulations in NetLogo. Two new undergraduate students currently work in its further evaluation as part of their student research projects. They simulate real world scenarios using FRB and BDI for modeling bank crises and their behavior over time. Their goal is to analyze the BDI library including the extension for efficiency in order to advance it. The project thereby evolves in the classroom, making the students the researches, developers and teacher at the same time. Constant communication between the students who take over the project and the supervising professor foster the undergraduate research and teaching.

VIII. Conclusions

This paper introduced the subject of an undergraduate student project that focused on extending a BDI library for NetLogo, by conducting a case study in the field of Agent-Based Computational Economics. The extension included a third mental notion to the agents, i.e., the addition of plans in order to allow the clustering of intentions and other plans for the modeling of agent's actions that consist of other actions, as well as the redefinition of intentions to allow calls by reference. Furthermore, this work presented how such undergraduate research supports AI classroom teaching at the Berlin School of Economics and Law. The idea to combine student research and student teaching has received a positive echo from all participating parties – the school, faculty and students.

Beyond the basics of mental notions, the BDI paradigm does not have fixed specifications dictating what the library should offer. Future work includes analyzing useful features and extending the planning of agents by adding dynamic prioritizing. Dynamic prioritizing would enable agents both to have and two swap competing plans at run time.

References

[1] U. Wilensky, "NetLogo," Evanston, IL, U.S.A., 1999, available online at http://ccl.northwestern.edu/netlogo/, retrieved January 3, 2013.

[2] I. Sakellariou, P. Kefalas, and I. Stamatopoulou, "Enhancing NetLogo to Simulate BDI Communicating Agents," in *Proceedings of the* 5^{th} *Hellenic Conference on Artificial Intelligence, SETN'08*, ser. Lecture Notes in Artificial Intelligence (LNAI), J. Darzentas, G. Vouros, S. Vosinakis, and A. Arnellos, Eds., vol. 5138. Syros, Greece: Springer Berlin Heidelberg, October 2008, pp. 263–275.

[3] M. E. Bratman, *Intention, Plans, and Practical Reason*. Cambridge, MA, U.S.A.: Harvard University Press, 1987.

[4] A. Rao and M. Georgeff, "Modeling Rational Agents within a BDI-Architecture," in *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning, KR'91*, J. Allen, R. Fikes, and E. Sandewall, Eds. Morgan Kaufmann, April 1991, pp. 473–484.

[5] ——, "BDI Agents: From Theory to Practice," in *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95*. San Francisco, CA, U.S.A.: AAAI Press / The MIT Press, June 12–14 1995, pp. 312–319.

[6] V. Mascardi, D. Demergasso, and D. Ancona, "Languages for Programming BDI-style Agents: an Overview," in *WOA*. Pitagora Editrice Bologna, 2005, pp. 9–15.

[7] L. Tesfatsion, "Agent-based Computational Economics: Modeling Economies as Complex Adaptive Systems," *Information Sciences*, vol. 149, pp. 263–269, 2003.

[8] L. Tesfatsion and K. Judd, Eds., *Handbook of Computational Economics: Agent-Based Computational Economics*, 1st ed. Elsevier, 2006, vol. 2.

[9] J. Arthur, W. Holl, B. Lebaron, R. Palmer, and P. Tayler, *Asset pricing under endogenous expectations in an artificial stock market*. Addison-Wesley, 1997, pp. 15–44.

[10] T. Klos and B. Nooteboom, "Agent-based computational transaction cost economics," *Journal of Economic Dynamics and Control*, vol. 25, no. 3-4, pp. 503–526, March 2001.

[11] M. Oeffner, "Agent-Based Keynesian Macroeconomics – An Evolutionary Model Embedded in an Agent-Based Computer Simulation," September 2008.

[12] M. Rothbard, *The Mistery of Banking*, Second ed. Ludwig von Mises Institute, 2008.

[13] J. Wiens, "Extending a NetLogo library for BDI-architectures in multi-agent systems," August 2012, computer Science Dept., Faculty of Cooperative Studies, Berlin School of Economics and Law, Student research paper, unpublished.

[14] M. Wooldridge, *An Introduction to MultiAgent Systems*, Second ed. UK: John Wiley & Sons Ltd, May 2009.

[15] "NetLogo Courses," Evanston, IL, U.S.A., n.d., available online at http://ccl.northwestern.edu/courses.shtml, retrieved January 3, 2013.

[16] D. Monett, R. Janisch, and S. Starroske, "NL-Analyzer: Enhancing Simulation Tools to Assist Multiagent Systems' Teaching," in *Proceedings of the Workshop Multi-Agent Systems for Education and Interactive Entertainment, MASEIE'2010*, W. van der Hoek, G. A. Kaminka, Y. Lesperance, M. Luck, and S. Sen, Eds., Toronto, Canada, May 10–14 2010, pp. 1–6.

[17] U. Wilensky, "Netlogo Wolf Sheep Predation model," Evanston, IL, U.S.A., 1997, available online at http://ccl.northwestern.edu/ netlogo/-models/WolfSheepPredation, retrieved January 3, 2013.

[18] D. Monett and M. Sänger, "Research and Teaching with Remo: Student research projects and teaching for and by undergraduate students," in *Proceedings of The 2012 International Conference on Frontiers in Education: Computer Science and Computer Engineering, FECS'2012*, H. Arabnia, V. Clincy, and L. Deligiannidis, Eds., vol. 2. Las Vegas, NV, U.S.A.: CSREA Press, July 2012, pp. 353–359.