Goal Oriented Agile Unified Process (Goaup): An Educational Case Study

Jun Lin[1,2]
[1]School of Computer Engineering, Nanyang Technological University, Singapore
[2]College of Software, Beihang University, 37 Xueyuan Rd., Beijing, China
e-mail: Jlin7@e.ntu.edu.sg; linjun@buaa.edu.cn

Chunyan Miao[1], Zhiqi Shen[1], Wei Sun[2]
[1]School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue,Singapore
e-mail: ascymiao@ntu.edu.sg; zqshen@ntu.edu.sg; weisun@buaa.edu.cn

Abstract—**We propose a novel goal-oriented method to model AUP software development process. Our method is based on Goal-Net modeling theory, which can be used to model complex process with phase goals and hierarchy goals. Our educational practices of Goal Oriented AUP (GOAUP) for Master of Software Engineering (MSE) students at College of Software, Beihang University showed that the work productivity and artifact quality can be improved by infusing GOAUP into their course and software development process. By analyzing and studying the educational case, our experiences that infusing GOAUP into the software engineering education are shared to professions and educators.**

Keywords-Agile Unified Process; Goal Net; Software Engineering Education

I. INTRODUCTION

As software development process (SDP) or software development life cycle (SDLC) is a dynamic, continuous, incremental, and chaotic process that is hard to be controlled. For assuring the quality of software produced from it, people have proposed a set of software engineering process models, which are given high hope to solve the software crisis since 1970s. At the beginning, the plan-driven methodologies, which focus on order and plan, such as waterfall model, V model, and spiral model etc., were well applied into large projects. However, excessive plan and order come with high cost, and even led to the inhibition of users' requirements and changes. After that, the research of use case-driven methodologies, which focus on user and iteration, such as United Software Development Process (USDP), Rational United Process (RUP) etc., and the research of process improvement methodologies, which focus on organization and team maturity, such as Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI) etc. became very active and popular in academia and industry. As those methodologies focus on users and team maturity, so they have been successful in industry for a long term. But both of them are still too planned and costly to small and medium projects. After year 2000, with the increasing number of development for social applications, mobile applications and cloud SaaS applications, Agile methodologies, such as Extreme Programming (XP), Scrum etc., have risen strong interests in academic and industry. As their lean, agile, and flexible characteristics are very suitable for responding to continuous changes and fast releases, especially in current turbulent economic environment.

The Agile software engineering or agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, face to face communication and encourages rapid and flexible response to changes. Today, more and more companies have embraced and joined into Agile, including almost all software giants such as Microsoft, Google, IBM, Facebook, SAP, Oracle, Salesforce and so on.

Sallyann and Helen listed the top 10 burning research issues in Agile Conference (XP) 2010 voted by more than 300 practitioners and researchers[1]. Most of them are challenges and potential directions for future agile software engineering research and education.

In our SE research and educational practice, we want to infuse the conceptual framework of agile software development into traditional methodologies, for example the Agile Unified Process (AUP[2]), a simplified version of the IBM Rational Unified Process (RUP[3], the best-known and extensively documented refinement of the Unified Software Development Process, USDP[4]), which describes a simple, easy to understand approach to develop business application software using agile methods and concepts yet still remaining true to the RUP, as shown in Fig. 1.

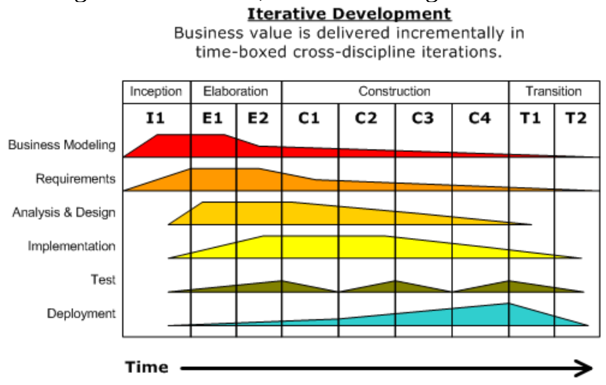

Figure 1. AUP, RUP, USDP all use incremental and iterative development model [5]

The AUP commonly applies agile methods including Test Driven Development (TDD), Agile Modeling (AM), Agile Change Management, and Database Refactoring to improve productivity [2].

Since 2005, College of Software at Beihang University has introduced RUP into practical course to train their team development skill for Master of Software Engineering (MSE). Later, in 2010 we started a new direction of Mobile & Cloud Computing for MSE, the first class has 94 full-time students who were trained by a tailored RUP during 2011 spring semester, and the second class (231 full-time students) and third class (176 part-time students) were trained by a tailored AUP in 2012 spring semester.

In this paper, we will report and sum up our research and educational experiences, including our modeling AUP via a Goal Oriented analysis method: Goal-Net modeling theory, and the horizontal and vertical comparison and analysis for our master students. The rest of this paper is organized as follows. In Section II, we first review existing work on the related fields, bringing the background of Agile Software Engineering and Goal-Net theory. In Section III we specifically describe our educational practice, result analysis, and the corresponding case study. The final section concludes this paper and discusses some future work.

II. Literature Review

A. *Agile Software Engineering*

Since 2001, the year of announcement of the agile manifesto, the research community has devoted a great deal of attention to agile methodologies. A literature search in the ISI Web of Science2 identified 1551 research papers that were published between 2001 and 2010 on agile software development [6].

During this period, Abrahamsson et al. (2002) [7], Cohen et al. (2004) [8], Erickson et al. (2005) [9], Dyba et al. (2008) [10], and Dingsoyr et al. (2010) [11] (2012) [6] gave the introductions to and overviews of agile methodologies respectively. These six reports describe the state-of-the-art and state-of-the-practice in terms of characteristics of the various agile methods, as well as lessons learned from applying such methods to industry at different stages.

Current Agile methodologies provide a conceptual framework that promotes foreseen interactions throughout the development cycle and process. The main features of them are iterative and incremental [12].

Agile methodologies include those methods more adaptive and active, which help software development to increase productivity and reduce risks. They are very effective where customer frequently changes the requirement. Since agile development has more iteration so developer can assure if a small modification meets customer's goal or not in the working software, better than one build system in the plan-driven process. It also involves more customer interaction and testing effort, it tries to satisfy the customer through early and continuous delivery of valuable software. This is useful when developer don't have a clear idea of the customer's goals. The development activities can be carried out using the iterative actions.

Agile methodologies attempt to provide many opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular cadences of work, known as sprints or iterations, at the end of which teams must present a shippable increment of work. Thus by focusing on the repetition of abbreviated work cycles as well as the functional product they yield. That's why agile methodologies could be described as "iterative" and "incremental". Fig. 2 shows an iterative development model.

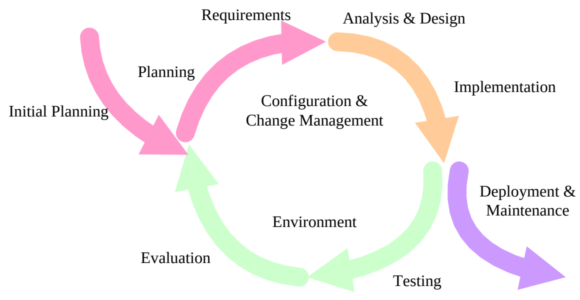

Figure 2. An iterative development model

In traditional waterfall development model or classical V model, development team only has one chance to get each aspect of the project right. But in an agile paradigm, every aspect of development, such as requirements, design, implementation, testing etc., is continually revisited throughout the lifecycle. When a team stops and re-evaluates the direction of a project every week or two weeks, there's always time to steer it into another direction.

The results of this "inspect-and-adapt" approach greatly reduce both development costs and time to market. Because a team's work cycle is limited to short time, it gives stakeholders recurring opportunities to calibrate goals and releases for success during the process. In essence, it could be said that the agile development methodology helps companies build the right product. Agile empowers teams to optimize their releases as it's developed, to be as competitive as possible in the marketplace.

In general, a typical agile software development process looks like Fig. 3.

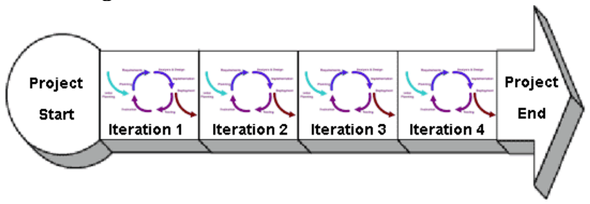

Figure 3. A typical agile software development process

As agile software development is just a conceptual framework for undertaking software engineering projects. So there are a number of specific agile methods espoused by the industry and Agile Alliance. According to the characteristics of agile, the following methods are generally considered as agile methods: [7]

Extreme Programming (XP) – a software development methodology which is intended to improve software quality and responsiveness to customer requirements changing. As a type of agile software development, it advocates frequent "releases" in short development cycles (time boxing), which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. [13]

Scrum – an iterative and incremental agile software development method for managing software projects and product or application development. Scrum has not only reinforced the interest in project management, but also challenged the conventional ideas about such management. Scrum focuses on project management institutions where it is difficult to plan ahead. Mechanisms of empirical process control, where feedback loops that constitute the core management technique are used as opposed to traditional command-and-control oriented management. It represents a radically new approach for planning and managing projects, bringing decision-making authority to the level of operation properties and certainties. [14]

Crystal Clear – a member of the Crystal family of methodologies as described by Alistair Cockburn and is considered an example of an agile or lightweight methodology. It can be applied to teams of up to 6 or 8 co-located developers working on systems that are not life-critical. The Crystal family of methodologies focuses on efficiency and habitability as components of project safety. Crystal Clear focuses on people, not processes or artifacts. [15]

Agile Modeling (AM) – a practice-based methodology for modeling and documentation of software-based systems. It is intended to be a collection of values, principles, and practices for modeling software that can be applied on a software development project in a more flexible manner than traditional modeling methods. [16]

Agile Unified Process (AUP) – a simplified version of the IBM Rational Unified Process (RUP) developed by Scott Ambler. It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the RUP. [2]

Dynamic Systems Development Method (DSDM) – an agile project delivery framework, primarily used as a software development method. First released in 1994, DSDM originally sought to provide some discipline to the rapid application development (RAD) method. In 2007 DSDM became a generic approach to project management and solution delivery. DSDM is an iterative and incremental approach that embraces principles of Agile development, including continuous user/customer involvement. [17]

Essential Unified Process (EssUP) – it was invented by Ivar Jacobson as an improvement on the Rational Unified Process. It identifies practices, such as use cases, iterative development, architecture driven development, team practices and process practices, which are borrowed from RUP, CMMI and agile development. The idea is that you can pick those practices that are applicable to your situation and combine them into your own process. This is considered an improvement with respect to RUP, because with RUP the practices are all intertwined and cannot be taken in isolation. [18]

Feature Driven Development (FDD) – an iterative and incremental software development process. It is one of a number of Agile methods for developing software and forms part of the Agile Alliance. FDD blends a number of industry-recognized best practices into a cohesive whole. These practices are all driven from a client-valued functionality (feature) perspective. Its main purpose is to deliver tangible, working software repeatedly in a timely manner. [19]

Kanban (development) – a method for developing software products and processes with an emphasis on just-in-time delivery while not overloading the software developers. It emphasizes that developers pull work from a queue, and the process, from definition of a task to its delivery to the customer, is displayed for participants to see. It can be divided into two parts: Kanban – a visual process management system that tells what to produce, when to produce it, and how much to produce, and the Kanban method – an approach to incremental, evolutionary process change for organizations. [20]

Lean Software Development (LSD) – a translation of lean manufacturing and lean IT principles and practices to the software development domain. Adapted from the Toyota Production System, a pro-lean subculture is emerging from within the Agile community. [21]

Open Unified Process (OpenUP) – a part of the Eclipse Process Framework (EPF), an open source process framework developed within the Eclipse Foundation. Its goals are to make it easy to adopt the core of the RUP/USDP. The OpenUP began with a donation to open source of process content known as the Basic Unified Process (BUP) by IBM. It was transitioned to the Eclipse Foundation in late 2005 and renamed OpenUP/Basic in early 2006. It is now known simply as OpenUP. [22]

Velocity Software Development (VSD) – a measure of productivity sometimes used in Agile software development. Velocity tracking is the act of measuring said velocity. The velocity is calculated by counting the number of units of work completed in a certain interval, determined at the start of the project. [23]

Adaptive Software Development (ASD) – focuses mainly on the problems in developing complex, large systems. The method strongly encourages incremental, iterative development, with constant prototyping. Fundamentally, ASD is about “balancing on the edge of chaos”; its aim is to provide a framework with enough guidance to prevent projects from falling into chaos, but not too much, which could suppress emergence and creativity. [19]

B. Goal-Net modeling method and theory

Goal-Net theory was proposed by Shen et al. in 2004 [24, 25], which is designed to model and design goal-oriented agents at first. Goal-Net model consists of four basic objects or concepts: states, transitions, arcs and branches. There are two types of states in Goal Net, composite state and atomic state. An atomic state accommodates a single state which cannot be split. A composite state, represented by a shadowed circle, represents a goal and may be split into sub states. States are interconnected by transitions. A transition primarily shows the relationship between the states it joins, specifying the task functions to be performed in a task list. Basically there are four kinds of relationships between two states, represented by transitions, including sequence, concurrency, choice, and synchronization. [25]

In 2004, Shen et al. presented Goal Net to model the goals of an agent and to model agent coordination in a multi-agent environment. Goal Net also serves as a practical methodology for engineering agent oriented software systems [24]. The next year, they refined the methodology for multi-agent system development. The new methodologies cover the whole life cycle of the agent system development, from requirement analysis, architecture design, and detailed design to implementation [26]. Based on this, in 2007, Yu et al. proposed a Goal Net Designer which is an integrated tool and Development Environment (IDE) for modeling agent behavior based on Goal Net model. The Goal Net Designer provides a way for users to simplify the various stages of agent design. It also can be used by the Multi-Agent Development Environment (MADE) automatically to create intelligent agents. [27]

In 2009, Zhang et al. proposed an agent planning system based on the Goal Net model. In their system, the agent's goals are identified and organized in a composite goal hierarchy. Three kinds of relations between goals are defined: choice, concurrency and synchronization. Actions between goals are designed to accomplish subsequent goals. The agent's desire is satisfied by accomplishing a serial of intermediary goals and finally achieving the ultimate goal that is satisfying the desire. The agent's action plan is a list of actions to accomplish the intermediary goals in the solution. Because Goal Net is designed by considering agent's possible desires directly, their works bridged the distance between BDI agent design and the planning system. They also proposed a searching algorithm to select goals in Goal Net. [28]

In 2010, Zhang et al. applied reinforcement learning algorithms for goal selection in a Goal Net to convert an original goal net to its counterpart that learning algorithm can operate on. They developed a reorganization algorithm to convert a refined goal net to a partially ordered network. The algorithm can convert concurrency and synchronization relationships to the choice relationship without losing any information in the original goal net. And then a reinforcement learning algorithm is applied to train the goal selection of the converted goal net. Their work showed that the goal net model can simulate motivated learning of goal selections. [29]

Goal-Net supports goal selection and action selection mechanism [28, 29]. Goal selection is used as the selection mechanism for choice relationship and is affected by some factors, such as achievement, cost, constraint and index etc. Action selection on the other hand provides sequential, rule-based or probabilistic inference execution for the tasks specified in a transition.

Goal-Net model provides a rich set of relationships and selection mechanism in providing a dynamic and highly autonomous agent problem-solving framework. Furthermore, a goal-oriented (GO) agent development methodology, namely GO methodology, based on Goal Net was also proposed in [26] by Shen et al. in 2005. GO methodology gives agent the ability to solve a problem by decomposing it into sub-goals. Sub-goals could be further decomposed until the hierarchical structure and the relationships of the goals are clearly defined. The temporal relationships and the transitions between the goals can be further identified. As a result, a Goal-Net model can be constructed that serves as the brain of an agent or an autonomous system, which enables the agent or system to select the next goal to achieve selected goal, as well as to select the next action to pursue the selected goal in a dynamic environment.

The research of Goal-Net theory is still ongoing. As a modeling method, it's a novel way to present the overview of system goals. Its goal selection and action selection mechanism also can provide flexibility to the path selection and optimization.

III. Method

During our preliminary research, the Goal-Net theory can be used to model the hierarchical goals in the agile software development process. We modeled the typical Agile United Process (AUP) or Scrum process as a Goal-Net diagram shown in Fig. 4, serves as the overview guide for students.

From Fig. 4, we can see that the top goal for one iteration is modeled as a top composite state named 'Software iteration finished'. To achieve the goal, we need to reach four sub-goals that are also composite states named 'Requirements Obtained', 'Design Finished', 'Implementation Finished', and 'Test Finished' sequentially. To achieve them, four lead transitions are required.

- Inception: the input transition for state of Requirements Obtained, which includes two atomic states: User stories obtained and Tasks obtained, and their three related transitions (tasks and conditions) shown in the figure. For Scrum, this transition can be executed in sprint planning activity.
- Elaboration: the input transition for state of Design Finished and output transition for state of Requirements Obtained, which includes three concurrent atomic states, their corresponding input transitions and one synchronized atomic state shown in the figure. For Scrum, this transition can be executed in daily scrum activity.
- Construction: the input transition for state of Implementation Finished and output transition for state of Design Finished, which includes three atomic states (Data structure obtained and Code obtained are concurrent, Code obtained and Unit test finished are sequential) and three corresponding input transitions. They are synchronized at a finished atomic state shown in the figure. For Scrum, this transition also can be executed in daily scrum activity.
- Transition: the input transition for state of Test Finished and output transition for state of Implementation Finished, which includes three

atomic states (Debug version obtained, Integration test version obtained and Working software obtained) and four corresponding input/output transitions. For Scrum, this transition also can be executed in daily scrum activity.

After these four goals are achieved, the AUP team can do some finishing work or Sprint retrospective activity for Scrum team to end this iteration.

There are two special atomic states shown at the bottom of figure 4, Bugs obtained and Test cases obtained, can cut across their parent composite states. In agile process, team member or tester can depict bugs or test cases anytime after user stories are obtained. Those bugs and test cases will be processed in transitions of Debugging and Acceptance testing respectively, which bring agility to the process.

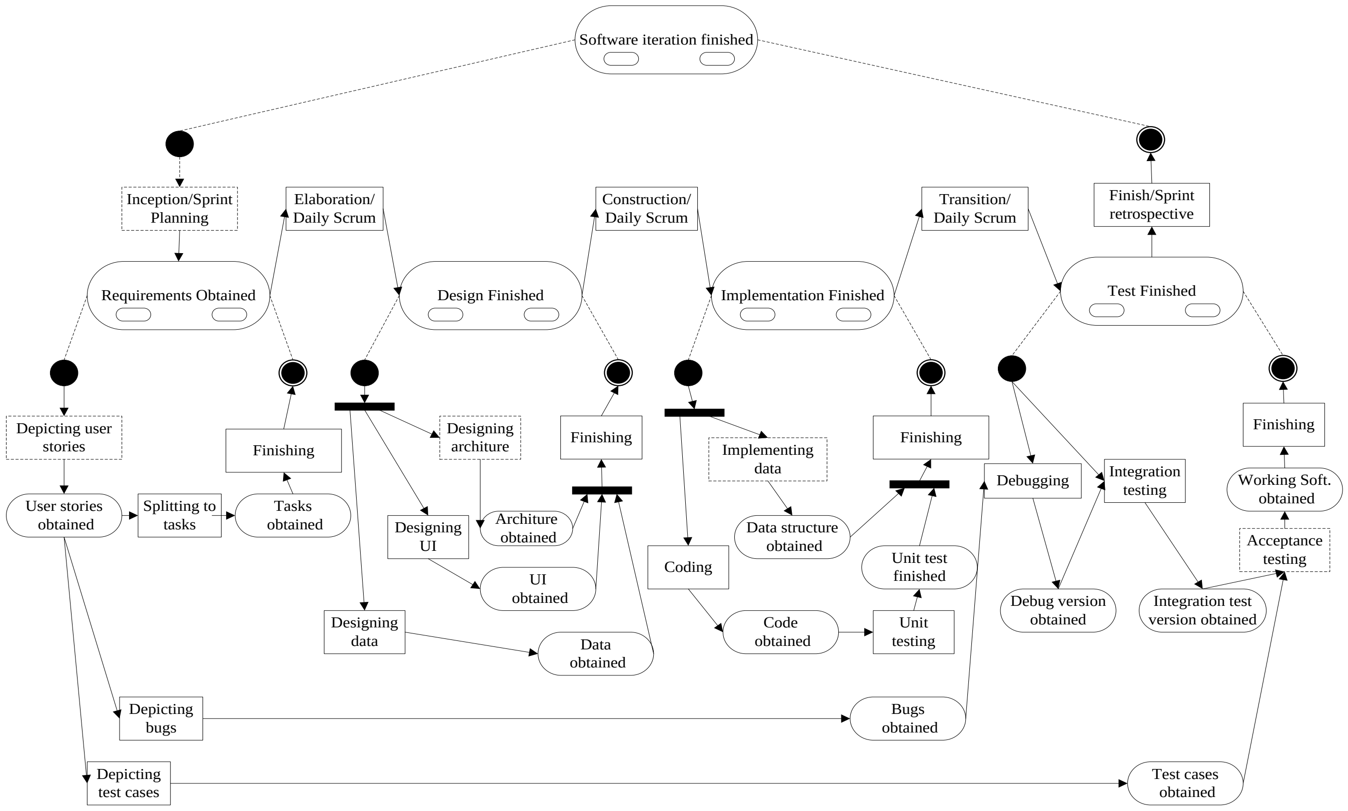

Figure 4. A tailored AUP/Scrum Goal-Net model for our course

IV. Practices & Case Study

As we mentioned before, the first MSE class on direction of Mobile & Cloud Computing has 94 full-time students who were trained by a tailored RUP during 2011 spring semester in College of Software (COS), Beihang University. They were spontaneously organized into 19 teams. The numbers of member in one team are strictly limited to 3-7 persons. Team members will take different roles in one team, such as leader/manager, analyst, designer, programmer, tester etc. The team leader should organize project meeting/activities regularly. Team can decide their project content by themselves, or they can pick up one of recommended projects from teachers. The reading material is the Rational Unified Process (version 2000). In 2012 spring semester, for the second class (231 full-time students) and third class (176 part-time students), the course requirements, team rules, execution mode, and evaluative criteria etc. were totally same with 2011 class. The difference is just some agile elements and features were introduced into the process as follows.

A. *The Roles in AUP Process*

Generally in an agile team, there are several roles, which have different names depending on the method listed above. Roles are not positions, any given person takes on one or more roles and can switch roles over time, and any given role may have zero or more people in it at any given point in a project.

Fig. 5 shows the overview structure of an agile team. The core agile team includes the team of developers who lead by the team lead, working closely with a product owner to build high-quality working software during the iterative and incremental process. Sometimes an architecture owner is also involved. The supporting casts including technical experts, domain experts and independent testers etc.

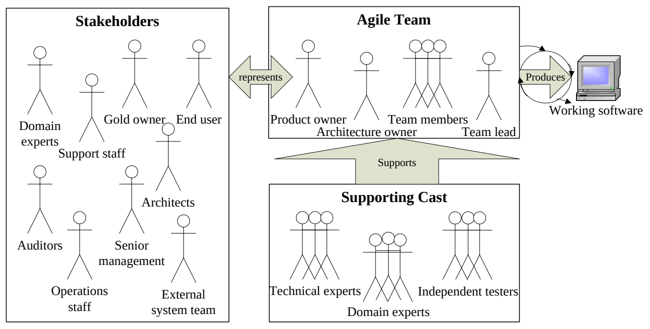

Figure 5. Organization structure of a typical agile team

From the organization structure, we can see the common agile roles include:

1) Core roles in agile team

- Team lead – the person whose role is responsible for facilitating the team, obtaining resources for it, and protecting it from problems. This role encompasses the soft skills of project management but not the technical ones such as planning and scheduling, activities which are better left to the team as a whole.
- Team member – the peoples, sometimes referred to as designer, tester, developer or programmer, whose role is responsible for the creation and delivery of a system. This includes designing, modeling, programming, testing, and release activities, and so on.
- Product owner – the product owner, called on-site customer in XP and active stakeholder in AM, represents the stakeholders. This is the one person whose role is responsible for a team (or sub-team for large projects). He/she is also responsible for the prioritized work item list (called a product backlog in Scrum), for making decisions in a timely manner, and for providing information in a timely manner.

2) Additional roles at scale

- Architecture owner – the person whose role is responsible for facilitating architectural decisions on a sub-team and is part of the architecture owner team which is responsible for overall architectural direction of the project.
- Stakeholder – a stakeholder is anyone who is a direct user, indirect user, manager of users, senior manager, operations staff member, the gold owner who funds the project, support IT staff member, auditors, program manager, developers working on other systems that integrate or interact with the one under development, or maintenance professionals potentially affected by the development and/or deployment of a software project.

3) Supporting roles at scale

- Technical experts – sometimes the agile team needs the help of technical experts, such as build masters to set up their build scripts or a DBA to help design and test their database. Technical experts are brought in on an as-needed, temporary basis, to help the team overcome a difficult problem and to transfer their skills to one or more developers on the team.
- Domain experts – as we can see in Fig. 5, the product owner represents a wide range of stakeholders, not just end users, and in practice it isn't reasonable to expect them to be experts at every specific domain. As a result the product owner will sometimes bring in domain experts to work with the team.
- Independent tester – effective agile teams sometimes need an independent test team working in parallel that validates their work throughout the lifecycle. This is an optional role, typically adopted only on very complex or big projects.

We require that the core roles must be involved in our student team and others are optional.

B. Requirement Management in AUP Process

Instead of other requirement analysis activities in most plan-driven process, agile team uses user stories to capture customer's goals, which are short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. User stories typically follow a simple template:

As a ***<type of user>****, I want to* ***<some goal>*** *so that* ***<some reason>****.*

User stories are often written on index cards or sticky notes, stored in a shoe box, and arranged on walls or tables to facilitate planning and discussion. Therefore, the team can strongly shift the focus from writing features to talking about them. In fact, these discussions are more important than whatever the text is written.

Product Owners are primarily responsible for user stories. But anyone else also can contribute to them. In actual environment many users write user stories. The first requirement may come from end user. The product owner, tech architect, scrum master, business analyst etc., anyone can update them but ultimately it is the product owner who is responsible for the backlog.

User stories should be written in a non-technical manner from the perspective of an end user. This user story will be further sliced. After fine tuning the stories to an extent this should be put to review to the agile team. The entire agile team should work on these stories to understand it perfectly. Any technical constraints or limitations should be noted down and presented to customer. Then finally those user stories will be stored in the product backlog and be divided into small piece of tasks to workers to implement. The product backlog is a prioritized list of functionalities that will be developed to the software product or service.

One of the benefits for agile user stories is that they can be written at varying levels of detail. We can write user stories that cover large amounts of functionality. These large user stories are generally known as epics. Here is an example epic from an online B2C marketplace product or services:

As a ***customer****, I want to* ***pay on mobile*** *so that* ***I can buy goods on mobile quickly****.*

As an epic is generally too large for an agile team to complete in one iteration, it needs to be split into multiple smaller user stories before it is worked on. The epic above might be split into dozens or more, including the following two:

As a ***VIP customer****, I want to* ***quickly pay by delivery*** *so that* ***I can buy goods on mobile quickly without paying immediately****.*

As a ***common customer****, I want to* ***quickly pay by credit card*** *so that* ***I can buy goods on mobile quickly****.*

Table I shows an example of user stories list in one of student project.

TABLE I. AN EXAMPLE OF USER STORIES LIST

ID	As a/an	I want to…	so that…
1	visitor	search goods online	I can find my favorite goods
1.1	visitor	search goods online by keyword	I can find my favorite goods according to typing a part of keyword
1.2	visitor	search goods online by category	I can find my favorite goods according to its category
2	visitor	sort the result after searching	I can find my favorite goods according to sorting result
2.1	visitor	sort the result by price	I can find my favorite goods according to price
2.2	visitor	sort the result by location	I can find my favorite goods according to location
2	customer	pay online	I can buy goods online
2.1	VIP customer	pay by delivery	I can buy goods online without paying immediately
2.2	common customer	pay by credit card	I can buy goods online
…	…	…	…

The split user stories then will be stored into backlog. There are four types of backlog in agile process (e.g. scrum) as follows:

- Backlog: a list of user stories, bugs and features that need to be handled.
- Product Backlog: a list of customer requirements for entire product.
- Release Backlog: a list of user stories, features and bugs that should be implemented in defined release.
- Iteration Backlog (Sprint Backlog): A list of user stories, features and bugs that should be implemented in defined iteration (e.g. one sprint in scrum).

C. Case Study and Analysis

1) Result Comparison

We did not require too much agile theories and methods for those student teams, but only encouraged them to equip above roles and requirement management methods into their development process. Table II shows the result summary comparisons between 2011 class and 2012 classes.

TABLE II. RESULT COMPARISONS BETWEEN 2011 AND 2012 LASSES

Graduate Students for MSE program	2011 Full-T	2012 Full-T	2012 Part-T
Number of teams	19	47	32
Project duration	3 months	2.5 months	1.6 months
Average team size	4.95	4.91	5.5
Average number of iteration meeting	4.89	3.06	2.69
Average number of artifact (Inception)	2	2.47	2.13
Average number of artifact (Elaboration)	1.84	2.06	2.41
Average number of artifact (Construction)	2.21	2.68	2
Average number of artifact (Transition)	1.53	1.32	1.53
Average quality of artifact (Inception) (5)	3.95	4.23	4.38
Average quality of artifact (Elaboration) (5)	4.11	4.13	4.28
Average quality of artifact (Construction) (5)	4.32	4.64	4.78
Average quality of artifact (Transition) (5)	4.21	3.74	3.53
Average final score of project (100)	83.05	85.4	88.25

From Table II, we can see the project durations for three classes are different. 2011 full-time class had 3 months to execute their projects; 2012 full-time class had 2.5 months; and 2012 part-time class had shortest time that is only 1.6 months. We hoped that infusing the agile thoughts can help them to speed up the project progress.

For comparing all aspects at a same baseline condition, we will take the project duration as the base for each class, and then other results will be divided by them.

2) Analysis and Lessons Learned

- Fig. 6 shows that as 2012 classes had shorter project development time, their average team size was bigger than 2011 class for increasing team man power when they spontaneously organized into teams, especially for 2012 part-time class. The average times of iteration (per month) for 2012 full-time class are less than 2011 full-time class for the time reason. However 2012 part-time class has more iteration times than others, as most of students in the class is working in software companies and has software development experiences.

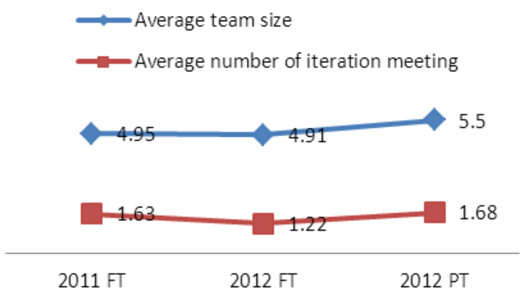

Figure 6. Comparison of team size and iteration times (per month)

- Fig. 7 shows that after unifying the comparison base to one month, the work productivity of 2012 classes are higher than 2011 class in all four AUP phases. Because of the rich work experiences for 2012 part-time students, they had highest productivity.

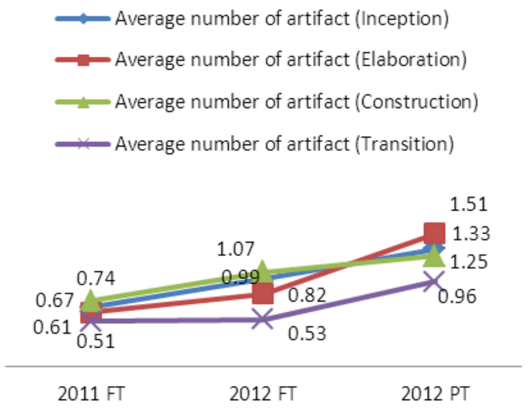

Figure 7. Productivity comparisons for different phase (per month)

- Fig. 8 shows the same result for the artifact quality in different AUP phases. The 2012 part-time class also had the highest quality.

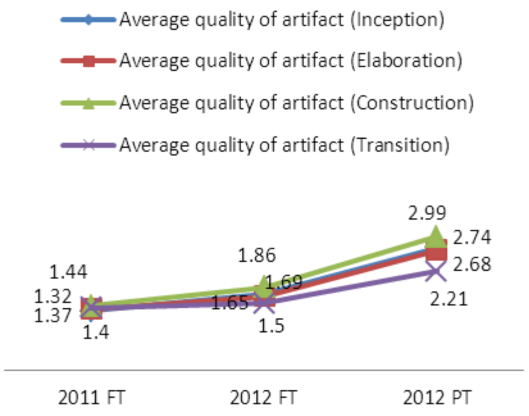

Figure 8. Artifact quality comparison for different phase (per month)

- The similar result also can be seen in Fig. 9. The 2012 part-time class had highest final score.

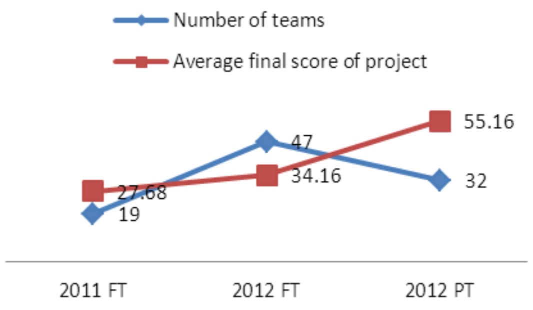

Figure 9. Final score comparison (per month)

V. Conclusions

In this paper, we propose a novel goal-oriented method to model AUP software development process based on Goal-Net modeling theory, which can be used on modeling complex process with phase goals and hierarchy goals.

The contributions in the paper include: 1) by modeling AUP via Goal-Net method, we have provided a fresh look and a new perspective to see AUP, 2) by analyzing and studying an educational case, we have shared our experiences of introducing Goal Oriented AUP (GOAUP) into the software engineering education to professions. Our educational practices of GOAUP for MSE students at College of Software, Beihang University showed that the work productivity and artifact quality can be improved by infusing GOAUP into their course and software development process.

Currently the GOAUP model just reflects overview of AUP from a Goal-Net view. With more detailed, hierarchy goals are discovered during software development process, how to monitor, manage and improve the agile process from the goal-oriented perspective can be a new research direction for future work.

References

[1] S. Freudenberg, and H. Sharp, "Currents Trends, People, Projects The Top 10 Burning Research Questions from Practitioners," Ieee Software, vol. 27, no. 5, pp. 8-9, Sep-Oct, 2010.

[2] I. T. Christou, S. T. Ponis, and E. Palaiologou, "Using the Agile Unified Process in Banking," Ieee Software, vol. 27, no. 3, pp. 72-79, May-Jun, 2010.

[3] J. Smith, "An introduction to the rational unified process," International Conference on Software Methods and Tools, Proceeding, pp. 263-263, 2000.

[4] I. Jacobson, G. Booch, and J. Rumbaugh, "The unified process (Reprinted from The Unified Software Development Process)," Ieee Software, vol. 16, no. 3, pp. 96-+, May-Jun, 1999.

[5] C. Larman, and V. R. Basili, "Iterative and incremental development: A brief history," Computer, vol. 36, no. 6, pp. 47-+, Jun, 2003.

[6] T. Dingsøyr, S. Nerur, V. Balijepally et al., "A decade of agile methodologies: Towards explaining agile software development," Journal of Systems and Software, vol. 85, no. 6, pp. 1213-1221, 2012.

[7] P. Abrahamsson, O. Salo, J. Ronkainen et al., Agile software development methods: review and analysis, 2002.

[8] D. Cohen, M. Lindvall, and P. Costa, "An introduction to agile methods," Advances in Computers, vol. 62, pp. 1-66, 2004.

[9] J. Erickson, K. Lyytinen, and K. Siau, “Agile modeling, agile software development, and extreme programming: The state of research,” Journal of Database Management, vol. 16, no. 4, pp. 88-100, Oct-Dec, 2005.

[10] T. Dyba, and T. Dingsoyr, “Empirical studies of agile software development: A systematic review,” Information and Software Technology, vol. 50, no. 9-10, pp. 833-859, Aug, 2008.

[11] T. Dingsoyr, T. Dyba, and N. B. Moe, “Agile Software Development: An Introduction and Overview,” Agile Software Development: Current Research and Future Directions, pp. 1-13, 2010.

[12] C. Larman, and V. R. Basili, “Iterative and incremental developments. a brief history,” Computer, vol. 36, no. 6, pp. 47 - 56, 2003.

[13] P. Abrahamsson, and J. Koskela, “Extreme programming: A survey of empirical data from a controlled case study,” in 2004 International Symposium on Empirical Software Engineering, Proceedings, 2004, pp. 73-82.

[14] O. Salo, and P. Abrahamsson, “Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum,” Iet Software, vol. 2, no. 1, pp. 58-64, Feb, 2008.

[15] A. Cockburn, Crystal Clear, A Human-Powered Methodology for Small Teams, 2004.

[16] H. Che, “Review of "Agile Modeling: Effective Practice for eXtreme Programming and the Unified Process by Scott W. Ambler", John Wiley & Sons, Inc, 2002, 0-471-20282-7,” Newsletter, ACM SIGSOFT Software Engineering Notes vol. 30, no. 4, pp. 83, 2005.

[17] A. Howard, “A new RAD-based approach to commercial information systems development: the dynamic system development method,” Industrial Management & Data Systems, vol. 97, no. 5-6, pp. 175-&, 1997.

[18] I. Jacobson, P. W. Ng, and I. Spence, “The essential unified process - A fresh start for process,” Dr Dobbs Journal, vol. 31, no. 9, pp. 40-+, Sep, 2006.

[19] A. F. Chowdhury, and M. N. Huda, “Comparison between Adaptive Software Development and Feature Driven Development,” 2011 International Conference on Computer Science and Network Technology (Iccsnt), Vols 1-4, pp. 363-367, 2012.

[20] M. Perona, and C. Benucci, “The Integrated Kanban System: A new software tool for kanban production,” It and Manufacturing Partnerships, vol. 7, pp. 75-92, 1996.

[21] M. Poppendieck, “Lean software development,” in 29th International Conference on Software Engineering: ICSE 2007 Companion Volume, Proceedings, 2007, pp. 165-166.

[22] A. Borg, M. Patel, and K. Sandahl, “Integrating an improvement model of handling capacity requirements with the OpenUP/Basic process,” Requirements Engineering: Foundation for Software Quality, vol. 4542, pp. 341-354, 2007.

[23] G. B. Alleman, M. Henderson, and R. Seggelke, “Making agile development work in a government contracting environment - Measuring velocity with earned value,” Proceedings of the Agile Development Conference, pp. 114-119, 2003.

[24] Zhiqi Shen, Chunyan Miao, Xuehong Tao et al., “Goal oriented modeling for intelligent software agents,” in Ieee/Wic/Acm International Conference on Intelligent Agent Technology, Proceedings, 2004, pp. 540-543.

[25] Zhiqi Shen, “Goal-oriented Modeling for Intelligent Agents and their Applications,” Nanyang Technological University, Singapore, 2005.

[26] Zhiqi Shen, D. T. Li, Chunyan Miao et al., “Goal-oriented methodology for agent system development,” in 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Proceedings, 2005, pp. 95-101.

[27] Han Yu, Zhiqi Shen, and Chunyan Miao, “Intelligent Software Agent Design Tool Using Goal Net Methodology,” in IAT '07. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2007., 2007, pp. 43-46.

[28] Huiliang Zhang, Zhiqi Shen, and Chunyan Miao, “Enabling Goal Oriented Action Planning with Goal Net,” in 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops, 2009, pp. 271-274.

[29] Huiliang Zhang, Zhiqi Shen, Chunyan Miao et al., “Motivated Learning for Goal Selection in Goal Nets,” in 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2010, pp. 252-255.