The 4th Kinneret Conference on SE Education

Creativity for the Software Engineer a novel Organic Paradigm with Applications

Zviel-Girshin Rina,
Ruppin Academic Center,
Israel
rinazg@ruppin.ac.il

An importance of Creativity

- The most valuable resource of the Software Engineer.
- Highest importance in our 21st century economy.
- The higher education institutions should prepare students to
  - creatively solve problems,
  - design and implement SE products.

Assumption

1. Students are highly motivated to learn and acquire creativity techniques.

2. The lion’s share of SE studies at all levels is dedicated to nurturing the creative side of the future Software Engineer.

IS THIS CORRECT?

A paradox

- In real life
  - the great value of creativity is the result of its inherent difficulty and rarity.
- But there is the rub.
  - The difficulty entails the rarity.

Teaching creativity

- Creativity is very difficult to teach and learn.
- The motivation of both student and teacher is less than desirable.
- In our study (qualitative research - 3 instructors, 25 students, quantitative research of 150 surveys)
  - creativity was graded as the most difficult and least motivating topic for the undergraduate SE student and teacher
    - (before that in K-12 studies).

OC paradigm

- An Organic Creativity (OC) paradigm.
- In very general terms we tried to analyze the basic problems
  - fear to err, writers block, difficulty to operate on a more abstract level, lack of experience and know-how and difficulty to formalize the creative process.
- The answer is to use a non-formal approach, though still scientific but taken from natural sciences.
- The Software Engineer should act less as mathematician and more as a doctor or an agronomist.

Principles of OC

- Dialectics
- Immersion in real life but with rigorous feedbacks and valid science
- Interactivity, learning and evolution mechanisms
- Concretizing the abstract
- A special creative life cycle
- Reflection and feedback
- Large DB and KB
- Need for constant learning from kindergarten to end of career.

A case study – OOP course

- Several assignment models were tested:

Single rigidly defined assignments

One large project – divided into parts

Several mini-projects

Single "loose" assignments

- However students found it very difficult to implement new cases/classes during the exam (quantitative research over 270 exams).

Flipped assignments

- "Flipped assignments" – testing some aspects of OC.

I WILL NOT FLIP THE CLASSROOM UPSIDE
I WILL NOT FLIP THE CLASSROOM UPSIDE DOWN

- Students had to define, design and implement several classes including inner objects, inheritance, polymorphism and more.

Students attitude towards OC

- Students
  - find this assignment to be very difficult and demanding during the course,
  - but report this assignment to be a good teaching method after the end of the course.
  - (qualitative research - 70 students from 5 different groups)

Results

- The use of OC environment improved dramatically the attitude towards creative aspects of OOP
    - (quantitative research over 120 exams).
- After several years of design and implementation, the OC paradigm evolved into useful technology environment
    - the heart of which is an Expert System enhanced with special mechanisms for learning and evolving.

Future plans

- An OC environment is not fully automated yet, it becomes more and more so, growing a much greater knowledge base.

- At this point the OC environment is ready to be deployed.

- Thanks:
  - Dr. Nathan Rosenberg, Paralex research,
  - Ruppin Junior College teaching instructors.