Eight steps for creating better bioinformatics software

Kristian Rother[1,2], Wojciech Potrzebowski[2], Tomasz Puton[1], Magdalena Rother[1], Ewa Wywiał[2], Janusz M. Bujnicki[1,2]
corresponding author: krother@genesilico.pl

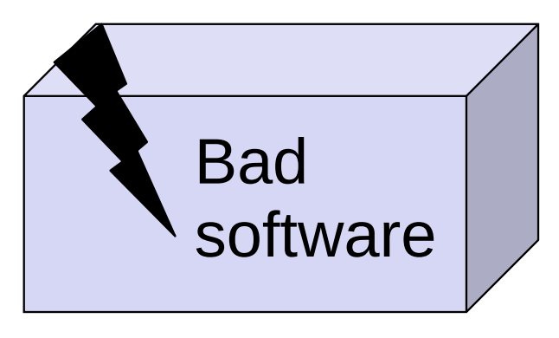
[1] Bioinformatics Laboratory, Institute for Molecular Biology and Biotechnology, Adam Mickiewicz University, Poznan, Poland
[2] Laboratory of Bioinformatics and Protein Engineering, International Institute of Molecular and Cell Biology, Warsaw, Poland

Problem: Lots of crappy programs

„It's almost finished."

„Not maintained any more."

„Doesn't compile on my machine."



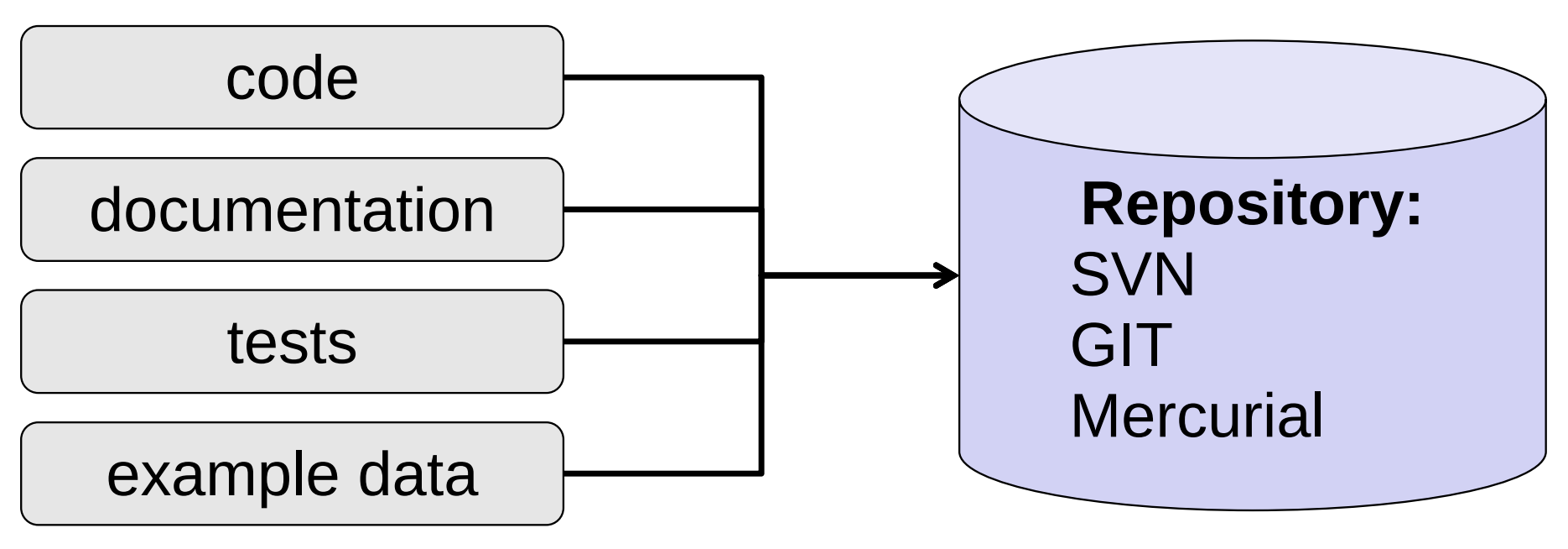
„The programmer has left the lab."

„This function is not documented."

„Doesn't work"

1. Use a repository

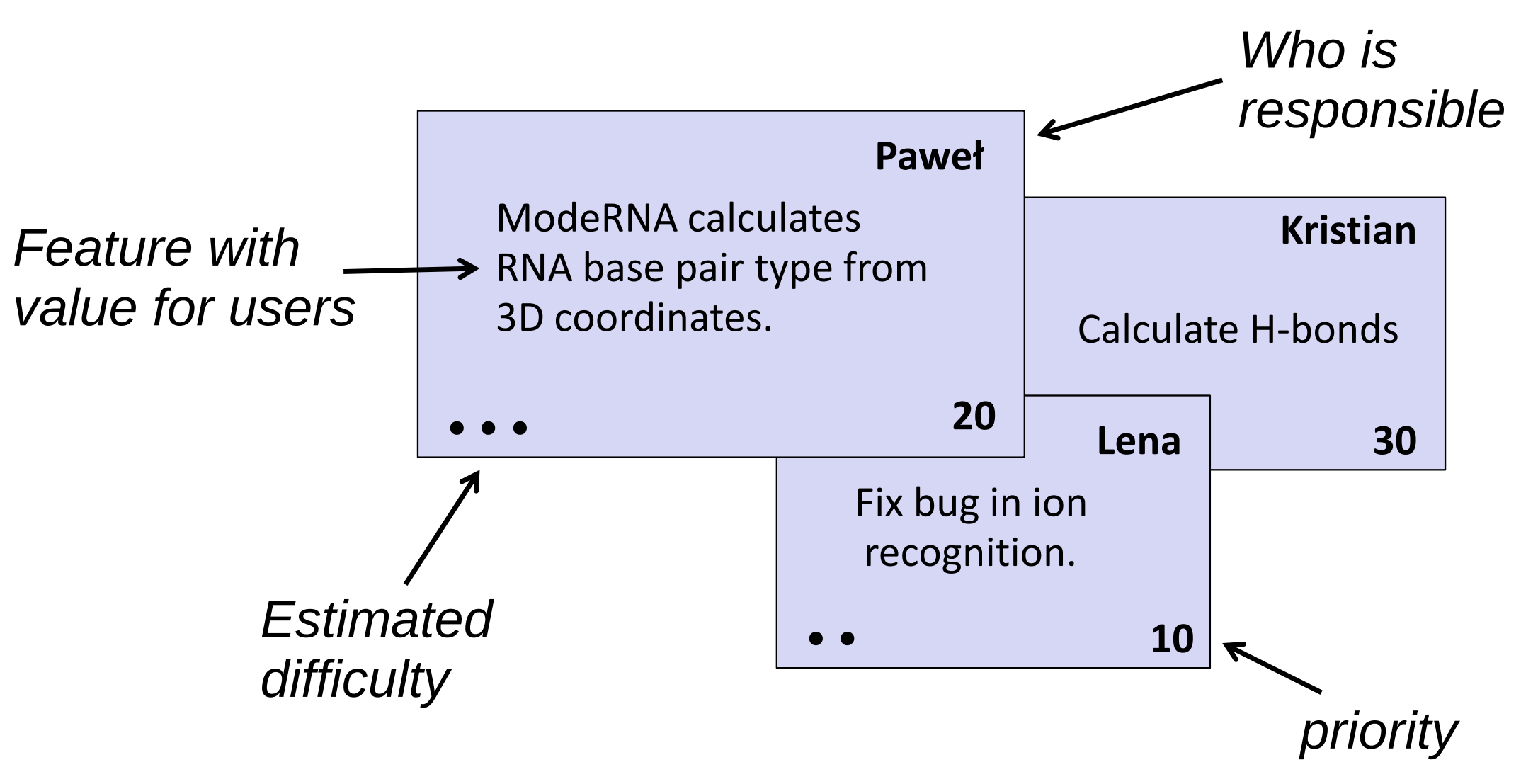
In a development team, a version control system or repository helps finding the latest edition of a file. (code, test data, configuration files, etc.). It also pays off as a backup system.



Case Study: RNA 3D Modeling

2. Planning with User Stories

Writing program features on cards helps to prioritize tasks, estimate effort, and reward oneself for achievements. The text on each card should convey some progress visible to the end user.


Who is responsible
Feature with value for users
Paweł
ModeRNA calculates RNA base pair type from 3D coordinates.
• • •
20
Kristian
Calculate H-bonds
30
Lena
Fix bug in ion recognition.
• •
10
Estimated difficulty
priority

3. Use Cases for complicated features

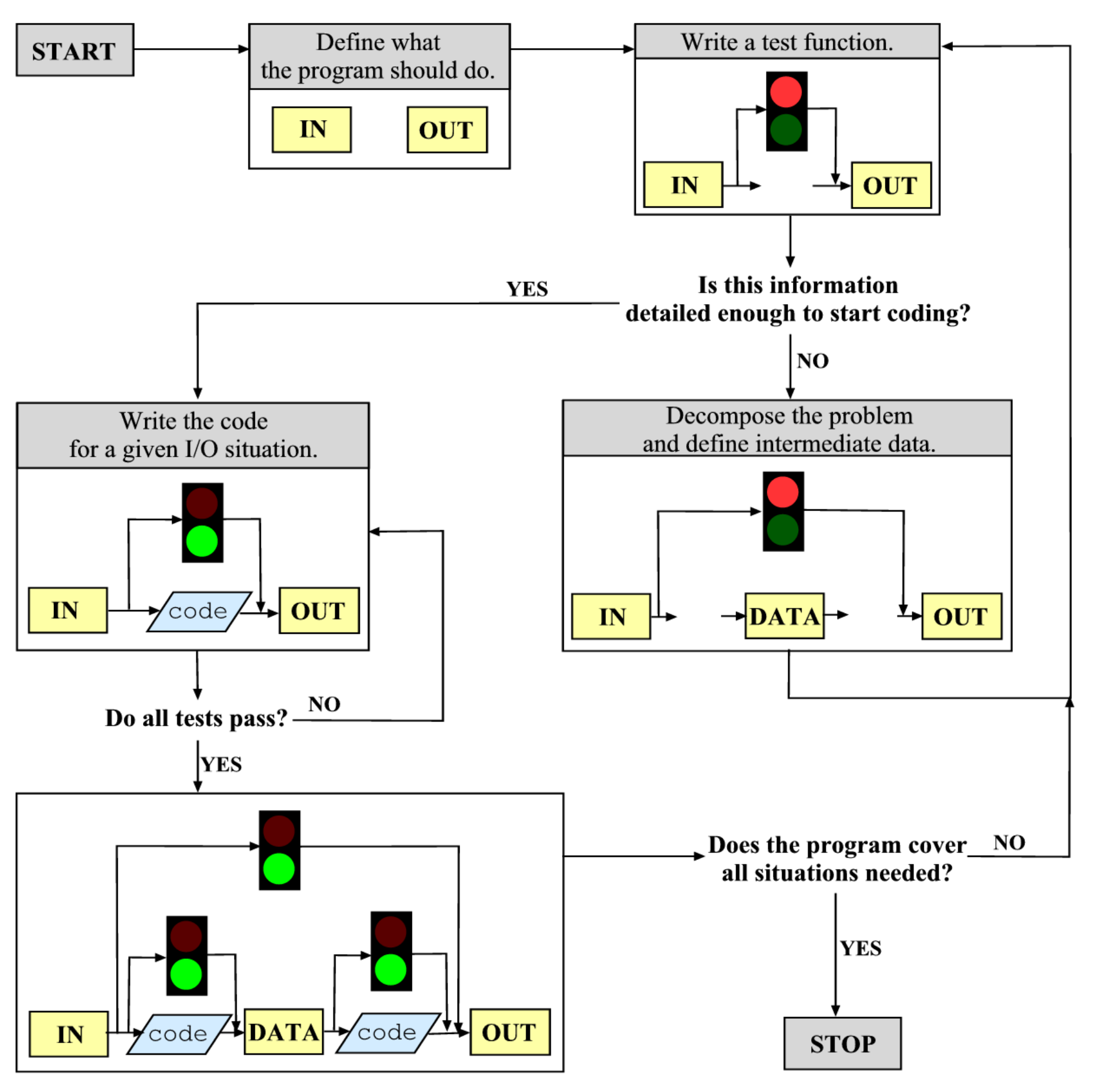
Use Case documents help formalizing complex tasks and communicating about them with users and other developers.

Use Case: Calculating base pairs

1. An user provides two nucleotide residues A,B.
2. The program checks whether A and B are close to each other.
3. The program calculates all hydrogen bonds between A and B.
4. If there are at least two hydrogen bonds:
   - G-C pairs with three bonds at the WC-edge are assigned the base pair type **+/+**.
   - A-U pairs with two bonds at the WC-edge are assigned the base pair type **-/-**.
   - All others are assigned a base pair type to their preferred contacting base edges (see Westhof 2001).
5. If there are less hydrogen bonds, the base pair type is identified as ***None***.

4. Write the test code first

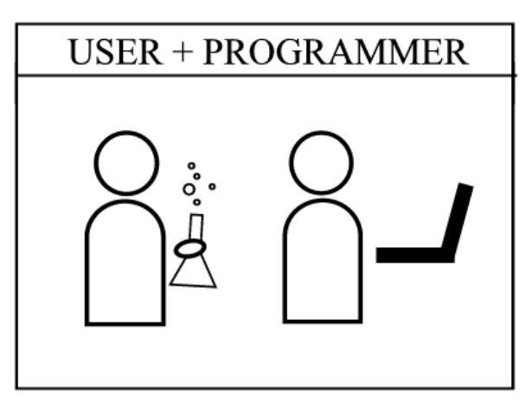
Test-Driven-Development (TDD) is a powerful method to guide the programming process. Writing tests forces a programmer to focus on what a program should do before working on the code itself. It also allows to measure progress.


START
Define what the program should do.
IN
OUT
Write a test function.
IN
OUT
Is this information detailed enough to start coding?
YES
NO
Write the code for a given I/O situation.
IN
code
OUT
Decompose the problem and define intermediate data.
IN
DATA
OUT
Do all tests pass?
NO
YES
IN
code
DATA
code
OUT
Does the program cover all situations needed?
NO
YES
STOP

5. Have clear responsibilities

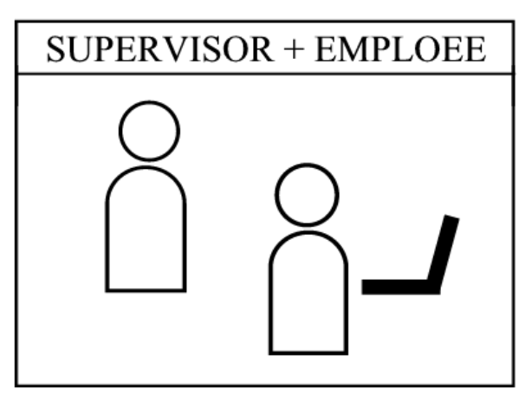
Social aspects are worth a consideration at the beginning. Standard roles can help to make sure that everybody knows who is responsible of which aspects of a project.

User:
•Describes the problem
•Provides example data
•Tests early versions



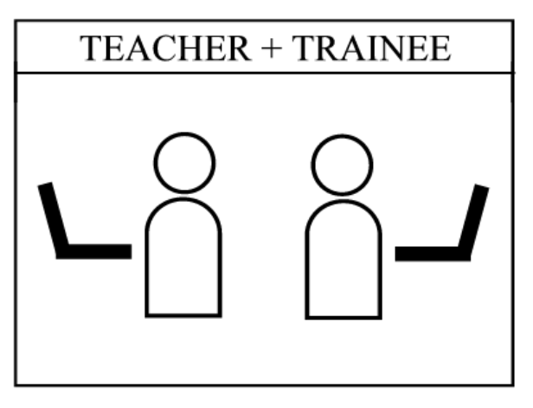
Programmer:
•Formalizes the problem
•Covers user requirements
•Writes the program

Supervisor:
•Defines the task
•Decides form of product
•Stays out of technical questions



Employee:
•Writes the program
•Solves technical problems

Teacher:
•Responsible of context
•Writes test functions
•Reviews the code

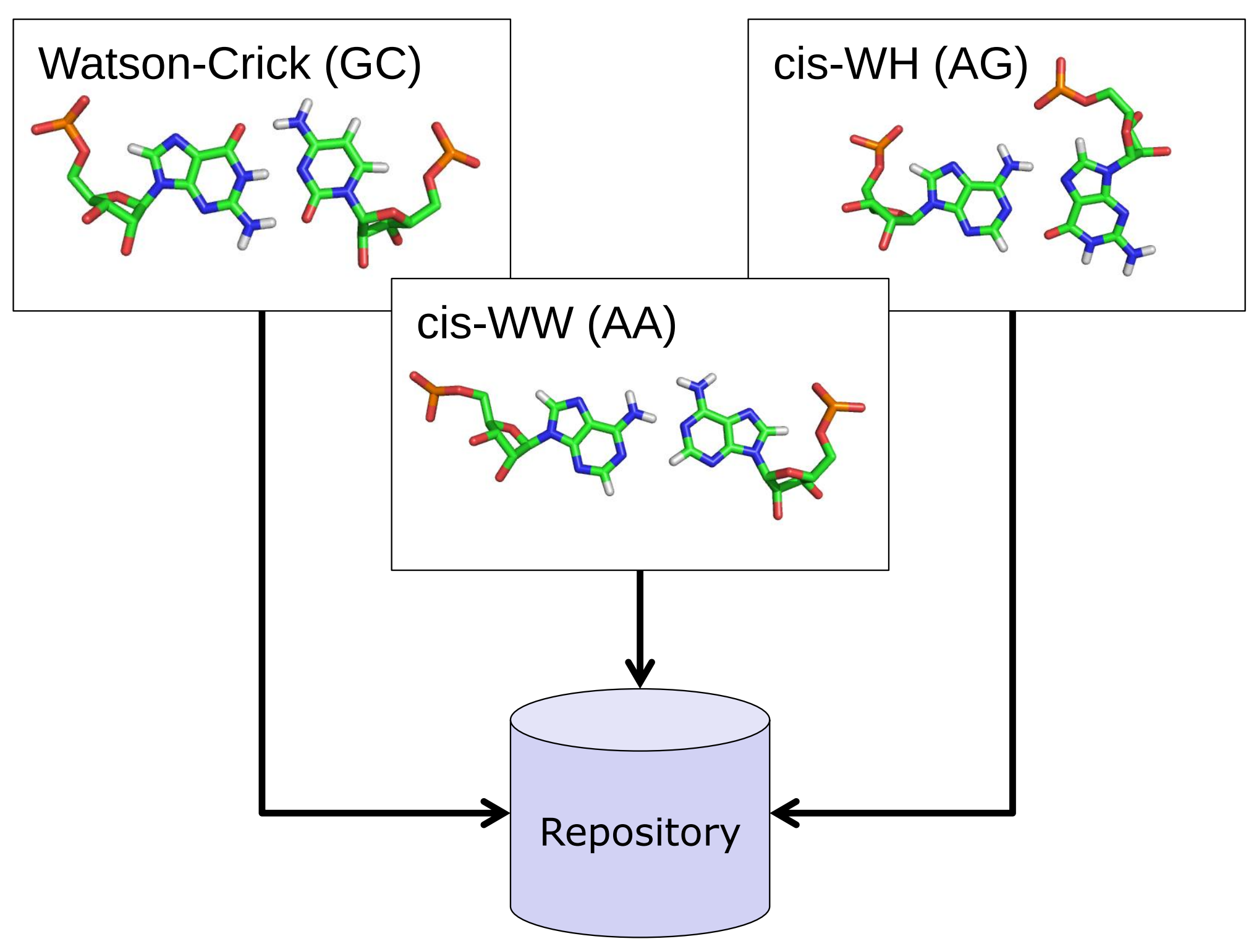


Trainee:
•Responsible of the code
•Makes sure all tests pass
•Reviews test functions.

Does everybody in your project agree on who is supposed to do what? Really everybody?

6. Collect example data

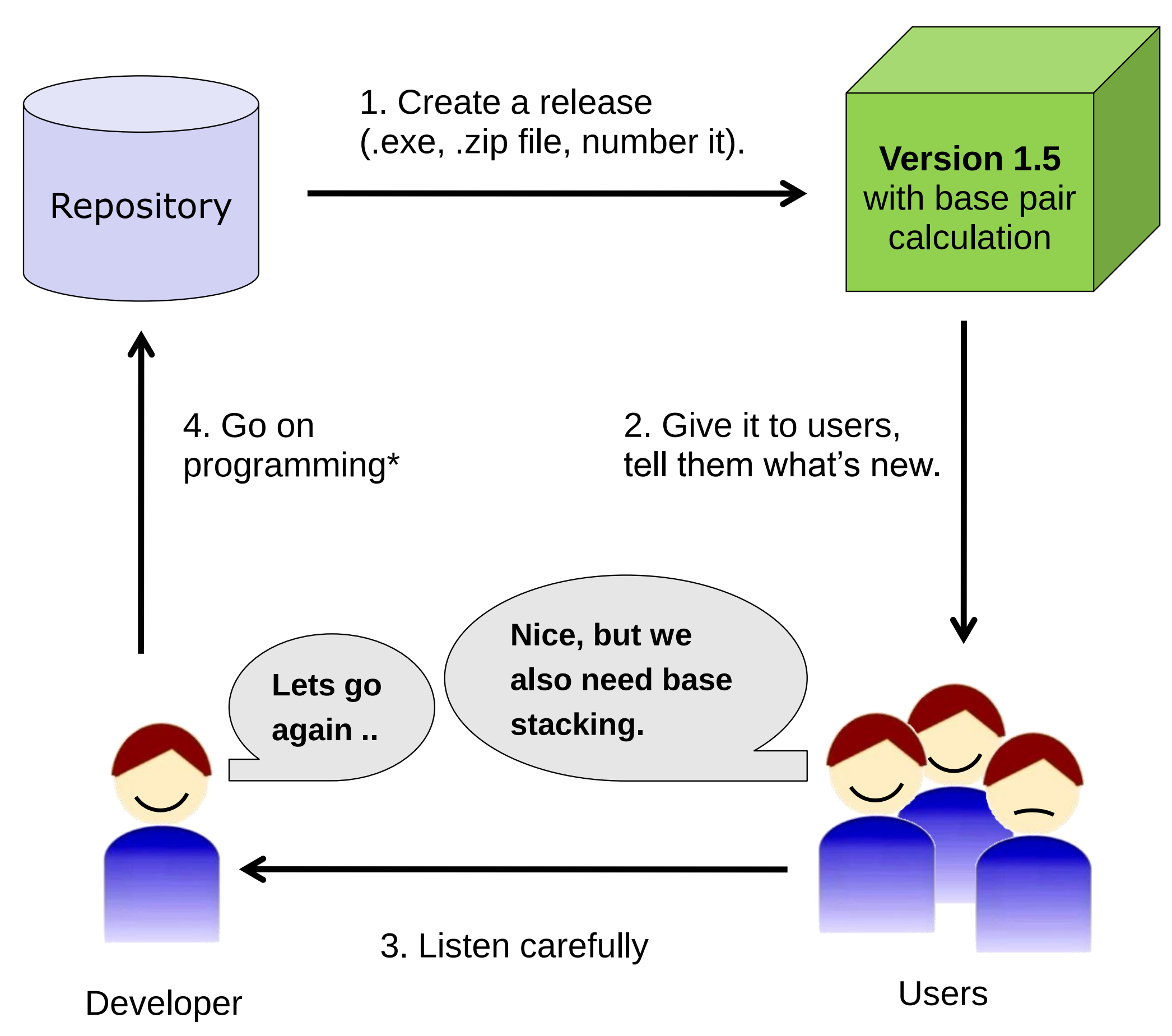
In most bioinformatics projects, it is essential to have a good set of exemplars for which you exactly know what output to expect from the program. More examples are necessary for border cases and unclean data – unless you want your users to find all of these.



Store example files in the repository

7. Release often

Making preliminary program versions frequently allows to collect constructive feedback – whether you are on the right track, and where the bugs are, and helps developers to decide what should be done next.


Repository
1. Create a release (.exe, .zip file, number it).
Version 1.5 with base pair calculation
2. Give it to users, tell them what's new.
Nice, but we also need base stacking.
Lets go again ..
3. Listen carefully
4. Go on programming*
Developer
Users

** If you think you can make money/fame/papers out of it.*

8. Reflect - Did it really work?

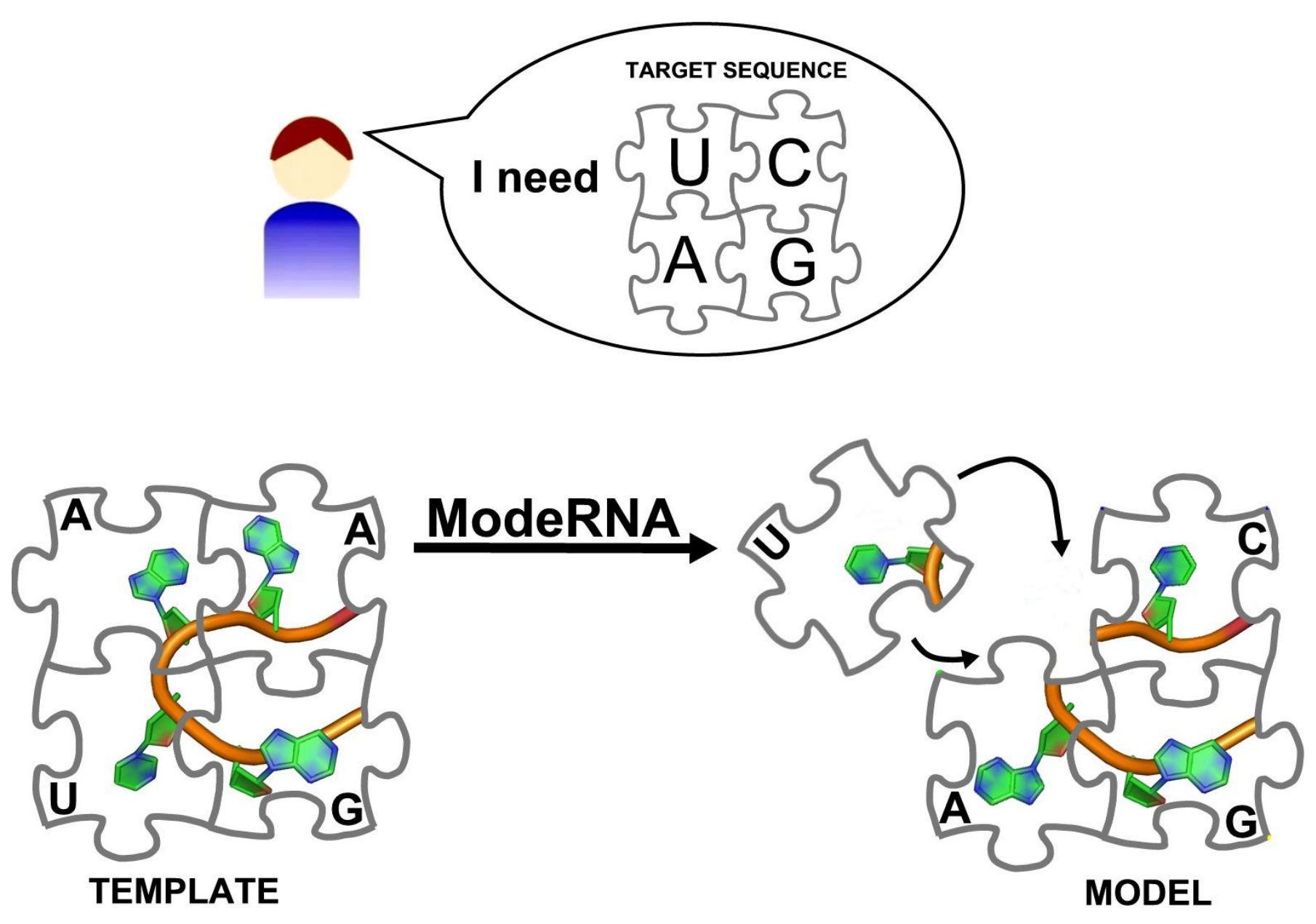
In our laboratory (www.genesilico.pl), 13 programming projects were examined, for the impact that use of these and other techniques have on successful completion.

PROJECT	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII
people	4	7	3	4	4	4	5	4	3	4	4	25	2
duration	6m	2y	2y	1y	6m	2y	3y	6m	1y	2y	2y	2y	6m
TECHNIQUES USED													
User Stories	•	○	○	•	•	•	○	○	○	○	•	•	○
Example data	•	○	•	○	•	○	○	•	○	○	•	•	○
UML	•	•	○	○	○	○	○	○	○	○	•	○	○
Repository	•	•	○	•	○	•	•	•	○	○	•	•	•
Ticket system	•	○	○	•	○	○	○	○	○	○	○	•	○
Code reviews	•	○	•	•	•	○	○	○	○	○	•	•	○
Unit tests	•	○	○	○	○	○	•	•	○	○	•	•	○
TDD	○	○	○	○	○	○	○	•	○	○	•	•	○
Pair programming	○	○	○	○	•	○	○	○	○	○	•	•	○
EVALUATION – survey with team members													
All features	•	○	•	•	•	○	•	•	○	•	•	○	○
On time	○	○	•	•	•	○	○	•	○	○	•	○	○
2+ releases	○	•	○	•	○	•	○	•	○	•	•	•	○
Still maintained	•	•	•	•	○	○	•	○	○	○	•	•	○
Would do it again	•	•	•	•	•	○	○	•	○	○	•	•	○

ModeRNA (project XI) is a toolkit for constructing RNA 3D structures. Available for download at http://iimcb.genesilico.pl/moderna.

Result: Better software

ModeRNA uses all techniques presented on this poster. In 2010, four releases of the program were completed. **See more on Poster 48.**



Acknowledgements

Supported by the DAAD (grant no.D/09/42768), by the Polish Ministry of Science and Higher Education (grants HISZPANIA/152/2006, POIG.02.03.00-00-003/09), the 6th+7th EU-FP (MRTN-CT-2005-019566, KSHG-CT-2005-518238) and AMU Poznan (PBWB-03/2009).