Information systems development methodology

version 0.1 – december 2004

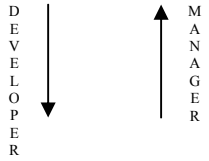
© Peter Peer, ppeer@ceit.es
CEIT, San Sebastian, Spain

for internal use only

GOAL:
The methodology assures unintrusive and simple way of formation of development documentation with minimal extra time investment!

METHODOLOGY – SUMMARY:
Level 1: uniform programming style
Level 2: versions management
Level 3: developers help formation
Level 4: use of XP method principles

reading direction



NOTE:
The methodology is concrete and consequently short and unambiguous. That is why it does not explain the terms used in this document.

LEVEL 1:

With uniform programming style we assure a very high level of readability:

1. Used tools:
   - MS Visual Studio 6.0 C++
   - VisualAssist plug-in
   - VAPlataforma code
   - Intel IPL and OpenCV libraries

2. Structure:
   - Subfunctional parts are appropriately enraptured (use of `tab`)
   - One row consists of only one program sentence
   - The sentence should be as short as possible; if it spans a lot over the width of the display then brake it in more parts
   - Divide names and operators with spaces
   - Methods should be as short as possible (divide into the smallest possible functional parts, put them in the same class)
   - Each class (`.cpp/.h`) put into a separate folder with class name

3. Naming convention:
   - Variables are presented with full names, consisting of only small letters and mark _ between the words
   - If we are dealing with a constant then the name must start with `c_`
   - Methods and classes are also named with full names, but each word starts with capital letter and there are no _ between the words
   - Always use the shortest possible full name
   - Dialog variables are named in the same manner, but additional information is in front of the name:
     a. Default suggestion by Visual Studio: `m_` (member of)
     b. Variable type: e.g. `m_b_` (for BOOL) or `m_cb_` (for CButton)
     c. The name that is appended is the same as what you appended to default suggestion by Visual Studio for Control ID (without the number, but with _): e.g. `IDC_CHECK_animate` and `m_b_animate`
     d. The appended names for all variables associated with the same Control ID are the same: `m_b_animate` and `m_cb_animate`

4. Language:
   - Names and comments are in English

5. Contract:
   - The contract (e.g. the comment above the method head/signature) should explain at least the return parameter and give the author
   - In `.h` (signature) file put each parameter in a separate row and if needed further explain its meaning; add the contract
   - This signature (with the contract) should be the same in `.cpp` file
   - The class (`.cpp/.h`) should start with the author(s) name(s)

6. Comments:
    - After the signature give a basic pseudo code outline of the procedure using natural language
    - In the code write comments only in parts that are harder to understand, i.e. that are more complex