PROJECT CAPACITY ASSESSMENT, FACTORS, COEFFICIENTS?

r – problem size (avg)

n – problem count

b – average salary

ε – difference of experience

M – employee count

t – resources (needed items)

w – risk

$$w = (\underbrace{e_w}_{\text{environment}} + \underbrace{e_t}_{\text{new technology}}) - e \leftarrow \text{existing solutions}$$

$$\$ = \{[(r \times n) \times b \cdot \varepsilon \cdot M] + t\} \cdot \underbrace{w}_{\text{risk}}$$

PROJECT CAPACITY ASSESSMENT, FACTORS, COEFFICIENTS?

<u>Factors</u>

1. Number of programmers
2. Work hours
3. Time taken to develop functional element in a code
4. Number of objectives

<u>Coefficient</u>

1. $\text{Efficiency of programmers} = \dfrac{\text{\#Senior developers}}{\text{\# Junior developers}}$
2. $\dfrac{\text{Lines of code per year}}{12} = \text{Lines of code / math}$

$$\frac{\text{Lines of code } \textit{per} \text{ month}}{\text{Number of programmers}}$$

PROJECT CAPACITY ASSESSMENT, FACTORS, COEFFICIENTS?

Overall time $\approx$

$$\approx \frac{\text{project amount}}{\text{amount of workforce}+\text{competence}+\text{arrangement of resources}}$$

amount of workforce = budget $\times$ amount of workspace + skill
(of workers)

project amount = complexity $\times$ cost of tasks $\times$ duration of task

CHECKLIST. SPECIFICATION STAGE TASKS?

- Write down the requirements
- Meet with the client
- Check if the requirements are achievable
- Separate the functional and non-functional requirements
- Setting a timescale
- Design prototype
- Risk management
- Make sure that the product is fully defined
- Consider the budget
- Ask the second opinion from someone
- Check the general standards
- Double check the checklist
- Consistency
- Plan next steps $\rightarrow$ think ahead

CHECKLIST. REALIZATION (PROG. , TEST.) STAGE TASKS?

- Choose programming method
  - Agile? $\leftarrow$ [soft. dev] $\rightarrow$ test driven?
- Discuss / collaborate with client
  - [what Prog. Lang. e^{+}c]
- Start documenting
  - [specifications etc.]
- Write the code + comment
- Create tests
  - [$\rightarrow$ comment the code]
- Test the code
- Review code
  - [debug], verify
- Create backups $\rightarrow$ GIT
- Create knowledge space
  - RTFM
- Continuous integration
- Userexp. testing
- Consult it lawyers
  - GbPR
- Review already existing functionality to use, not create
- Look out for dependencies
  - If something small makes it crash, discard
- Meet up regularly with team
- Keep notes of progress
- Create checklist

CHECKLIST. MAINTENANCE STAGE TASKS?

1. Does the customer understand the program?
2. Does the program have any bugs / errors / unpredictable event?
3. Does the software run in an intended environment?
4. Does the software require adding any new functionality?
5. Does the software work according the reasonable time?
6. Is the customer service capable of dealing with problems?
7. Does the software meets all expectations of customer?
8. Is the software scalable?
9. Is the software fault tolerable?
10. Can we apply metrics analysis?
11. Is the bug reproducable?
12. Can the software up to date?
13. Is the documentation enough for users?