Gauge for Programmer

Elaboration of software developers evaluation system – theory and practice

Valentin Anoprenko
VP of R&D
anoprenko@devexperts.com

What are we going to talk about

Evaluation of relative efficiency/value of a developer from employer's (company) standpoint in longterm

What we are *not* going to talk about

- **Evaluation of developer's effort in short term (BSC/KPI, bonuses, etc)**
- **IT labor-market and its influence on developers evaluation, salaries, etc**

Why to gauge?

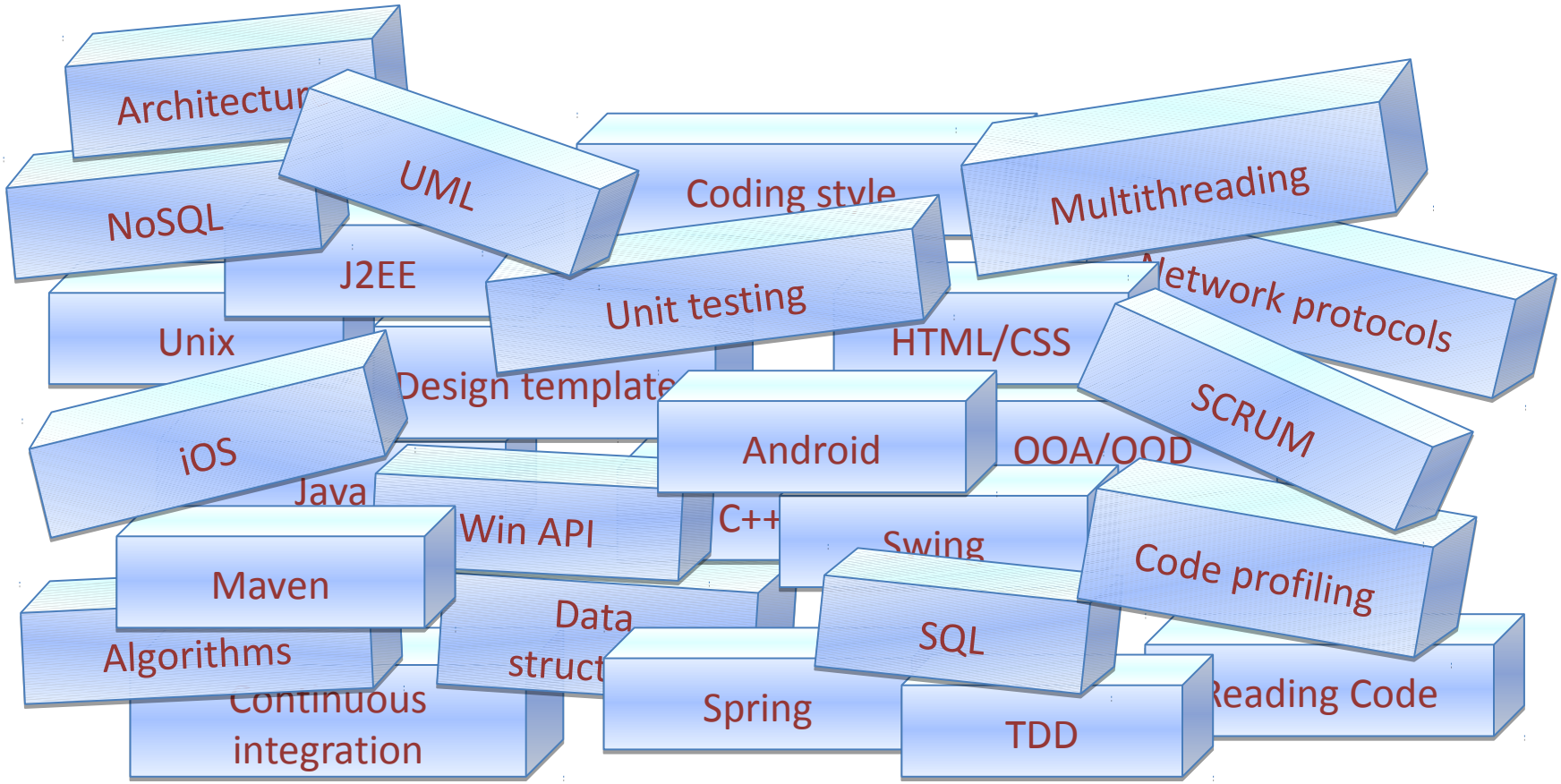
- Fair compensation package
- Directions of professional growth
- Career promotion

Required gauge properties

- **Objective**
- **Comprehensive**
- **Robust**
- **Comparable**

Implied obstacles

Too many factors!



Implied obstacles

There are “immeasurable factors”!

- **Creativity**
- **Soft skills**
- **Common sense**
- **Ability to solve complex problems**
- **Responsibility**

Implied obstacles

No common scale

- Different evaluation criteria
- Different criteria “weight”
- Different project/team needs

Implied obstacles

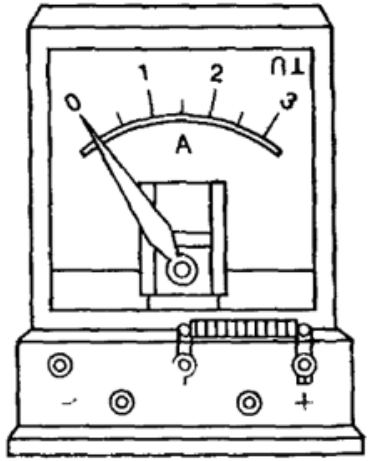
Subjective assessment

- Common impression instead of facts
- «Administrative rent»
- Influence of others’ opinion

Possible approaches

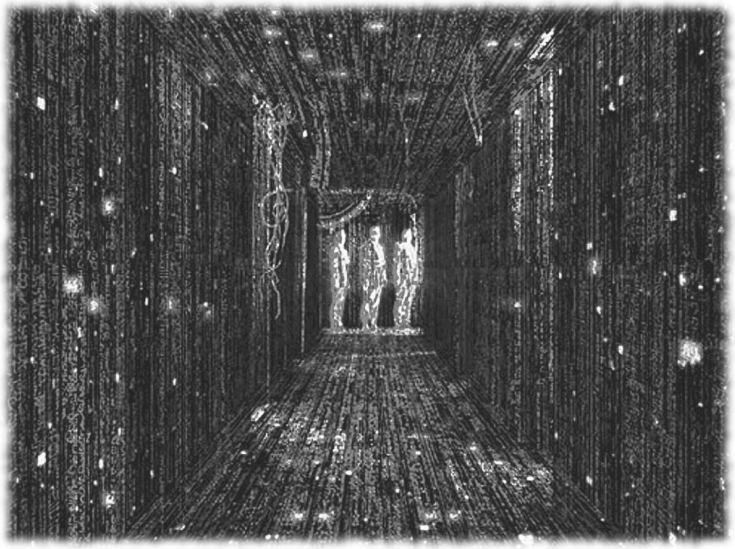
Measuring the results via number of

- lines of code written
- functional points added
- story points burned
- new features implemented
- defects added
- etc



Possible approaches

Programmer competency matrix



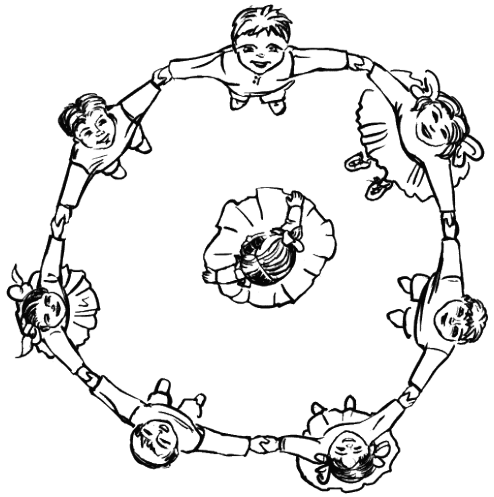
Programmer competency matrix

	2n (Level 0)	n2 (Level 1)	n (Level 2)	log(n) (Level 3)
Computer Science				
data structures	Doesn't know the difference between Array and LinkedList	Able to explain and use Arrays, LinkedLists, Dictionaries etc in practical programming tasks	Knows space and time tradeoffs of the basic data structures, Arrays vs LinkedLists, Able to explain how hashtables can be implemented and can handle collisions, Priority queues and ways to implement them etc.	Knowledge of advanced data structures like B-trees, binomial and fibonacci heaps, AVL/Red Black trees, Splay Trees, Skip Lists, tries etc.
algorithms	Unable to find the average of numbers in an array (It's hard to believe but I've interviewed such candidates)	Basic sorting, searching and data structure traversal and retrieval algorithms	Tree, Graph, simple greedy and divide and conquer algorithms, is able to understand the relevance of the levels of this matrix.	Able to recognize and code dynamic programming solutions, good knowledge of graph algorithms, good knowledge of numerical computation algorithms, able to identify NP problems etc.
systems programming	Doesn't know what a compiler, linker or interpreter is	Basic understanding of compilers, linker and interpreters. Understands what assembly code is and how things work at the hardware level. Some knowledge of virtual memory and paging.	Understands kernel mode vs. user mode, multi-threading, synchronization primitives and how they're implemented, able to read assembly code. Understands how networks work, understanding of network protocols and socket level programming.	Understands the entire programming stack, hardware (CPU + Memory + Cache + Interrupts + microcode), binary code, assembly, static and dynamic linking, compilation, interpretation, JIT compilation, garbage collection, heap, stack, memory addressing…
Software Engineering				
source code version control	Folder backups by date	VSS and beginning CVS/SVN user	Proficient in using CVS and SVN features. Knows how to branch and merge, use patches setup repository properties etc.	Knowledge of distributed VCS systems. Has tried out Bzr/Mercurial/Darcs/Git
build automation	Only knows how to build from IDE	Knows how to build the system from the command line	Can setup a script to build the basic system	Can setup a script to build the system and also documentation, installers, generate release notes and tag the code in source control
automated testing	Thinks that all testing is the job of the tester	Has written automated unit tests and comes up with good unit test cases for the code that is being written	Has written code in TDD manner	Understands and is able to setup automated functional, load/performance and UI tests
Programming				
problem decomposition	Only straight line code with copy paste for reuse	Able to break up problem into multiple functions	Able to come up with reusable functions/objects that solve the overall problem	Use of appropriate data structures and algorithms and comes up with generic/object-oriented code that encapsulate aspects of the problem that are subject to change.

Possible approaches

«360º feedback»

- peers
- supervisor(s)
- subordinates
- self-evaluation



Possible approaches

Job evaluation methods (point factor analysis)

- Hay Guide Charts
- Mercer's International Position Evaluation System

Hay method

- Job performance evaluation methodology
- Allows evaluation of creative jobs
- Introduced in 1950s by Edward N. Hay
- Owned and distributed by “Hay Group” consulting company
- Used by 8000+ organizations across the world

Hay method

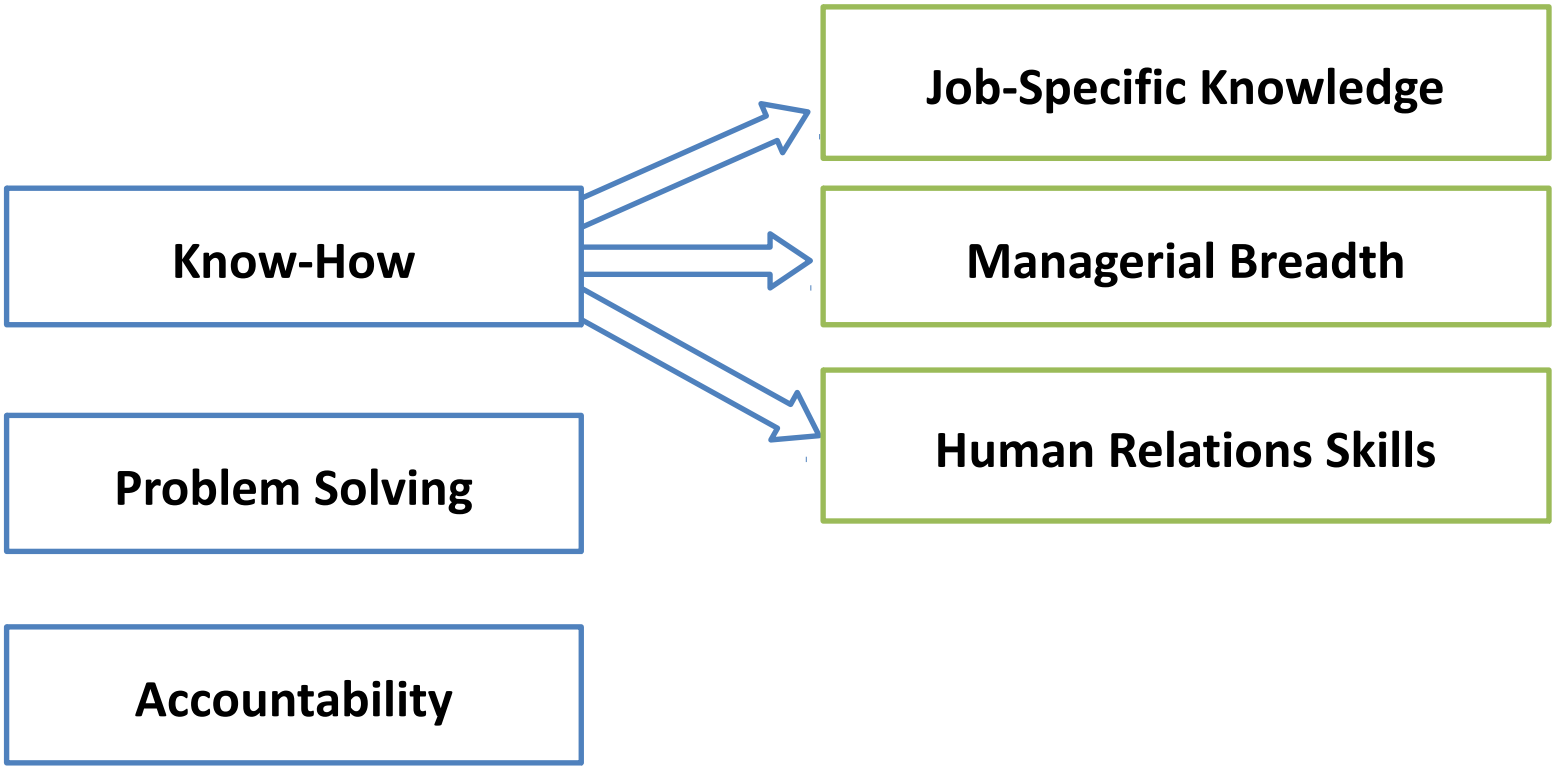
Based on 3 major evaluation factors

Know-How
(knowledge and skills)

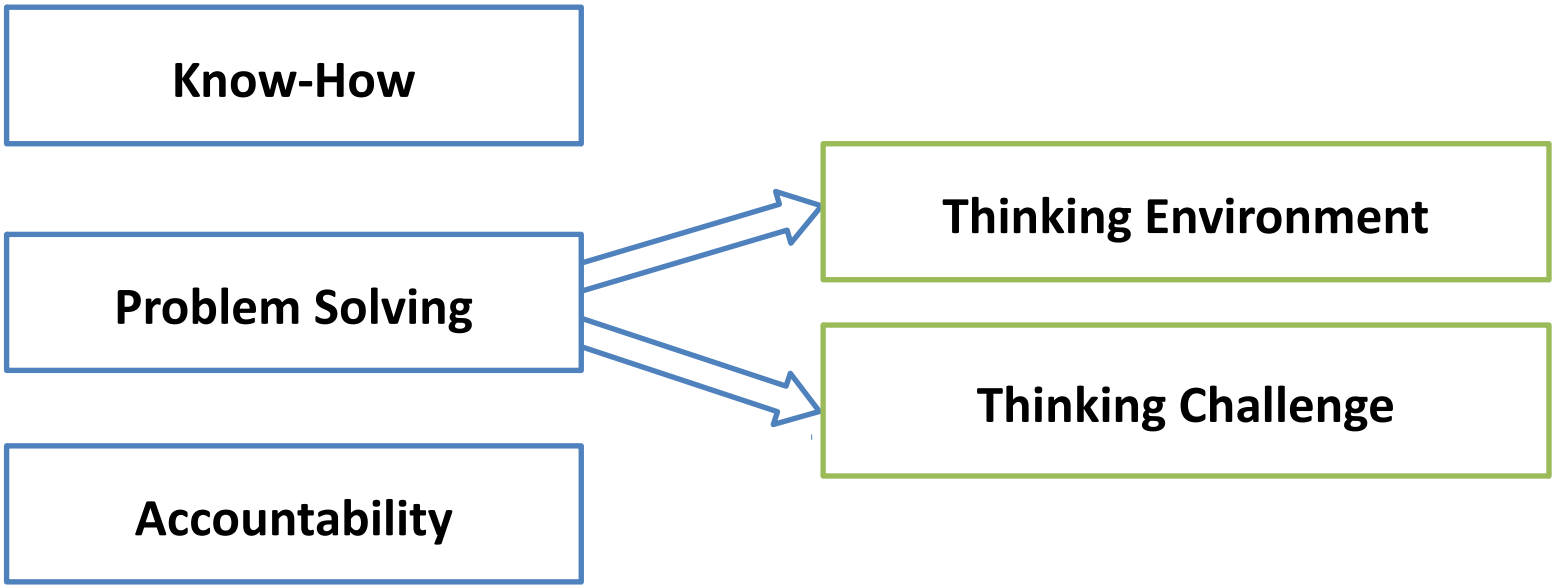
Problem Solving

Accountability

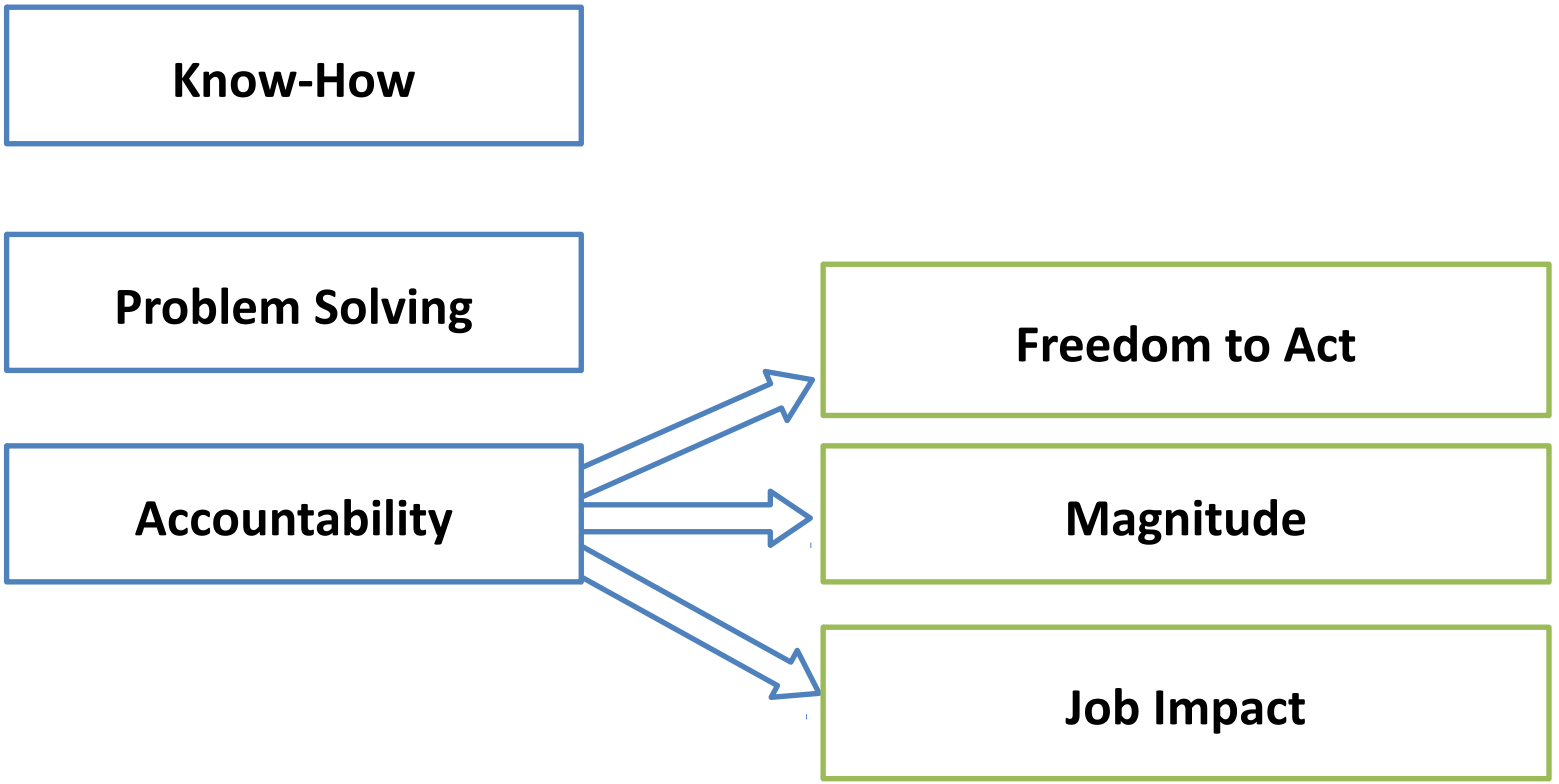
Hay method


Know-How
Problem Solving
Accountability
Job-Specific Knowledge
Managerial Breadth
Human Relations Skills

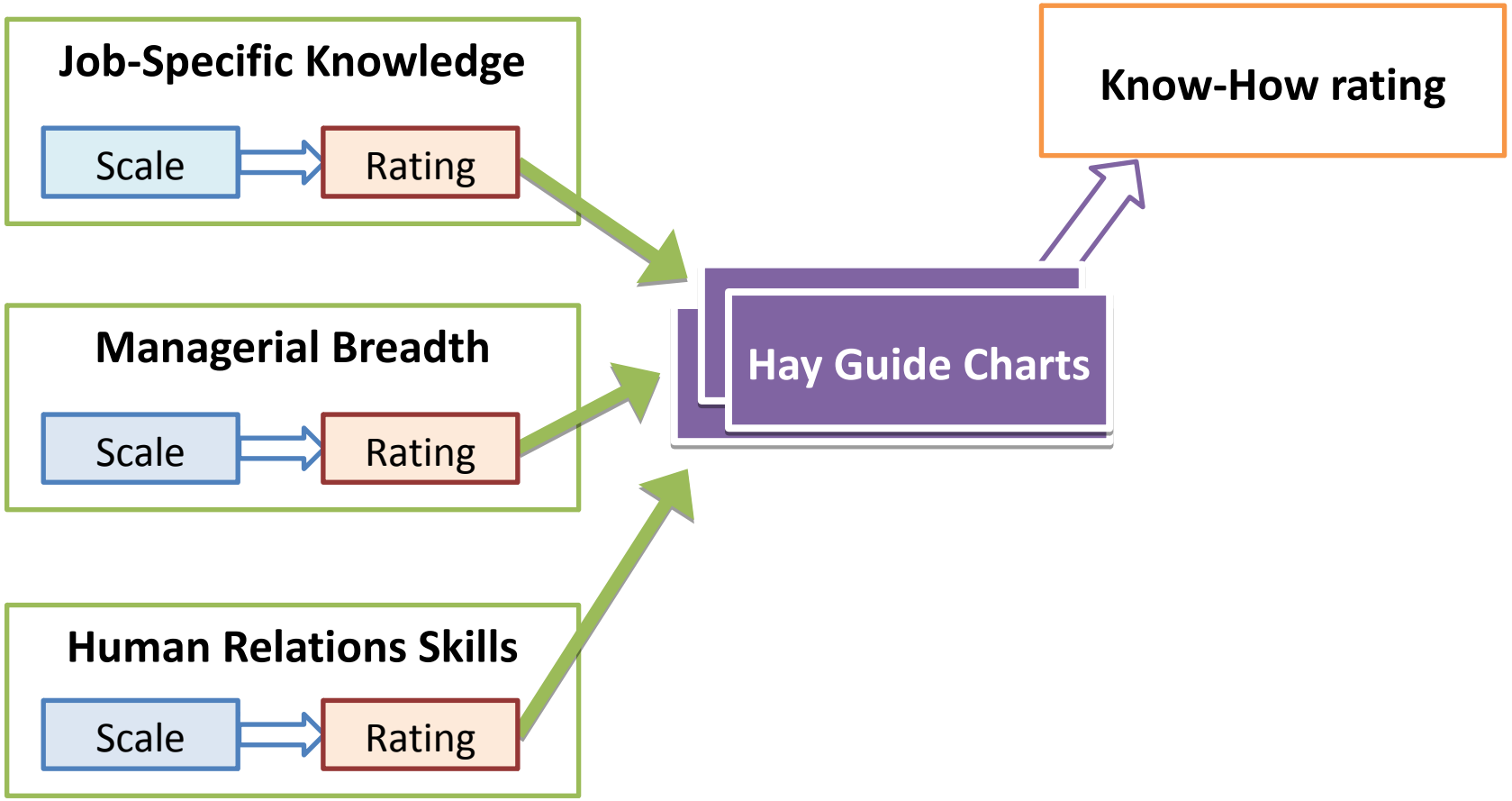
Hay method


Know-How
Problem Solving
Accountability
Thinking Environment
Thinking Challenge

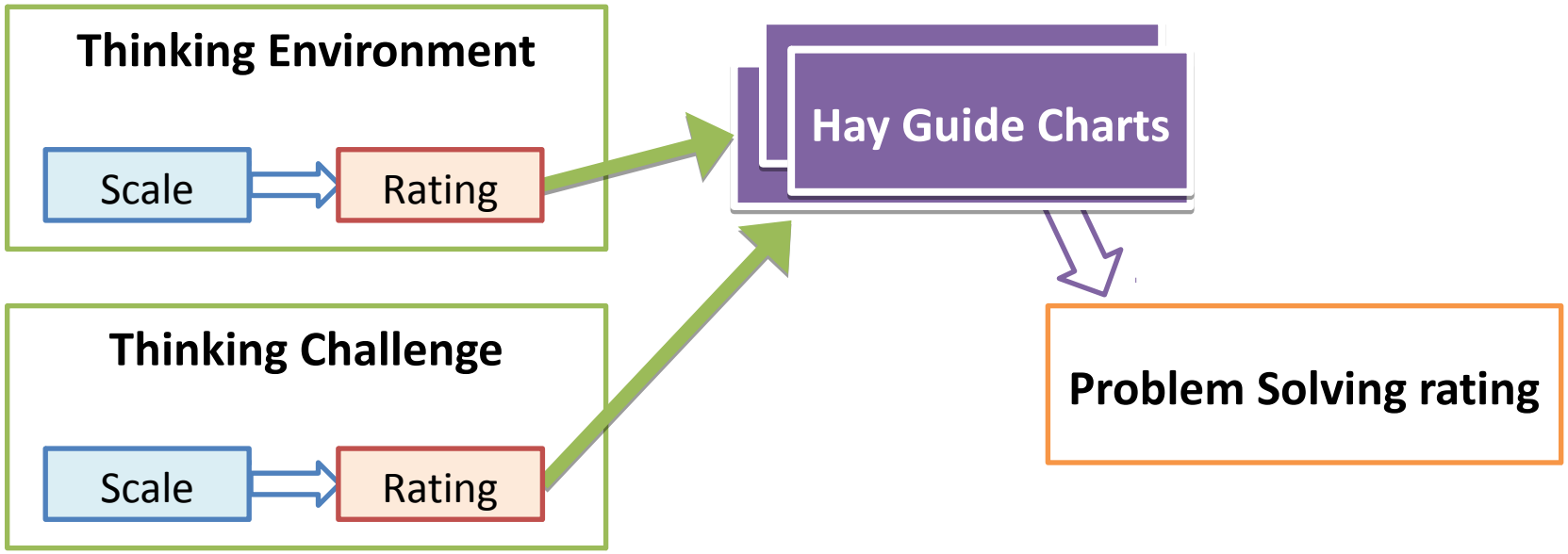
Hay method


Know-How
Problem Solving
Accountability
Freedom to Act
Magnitude
Job Impact

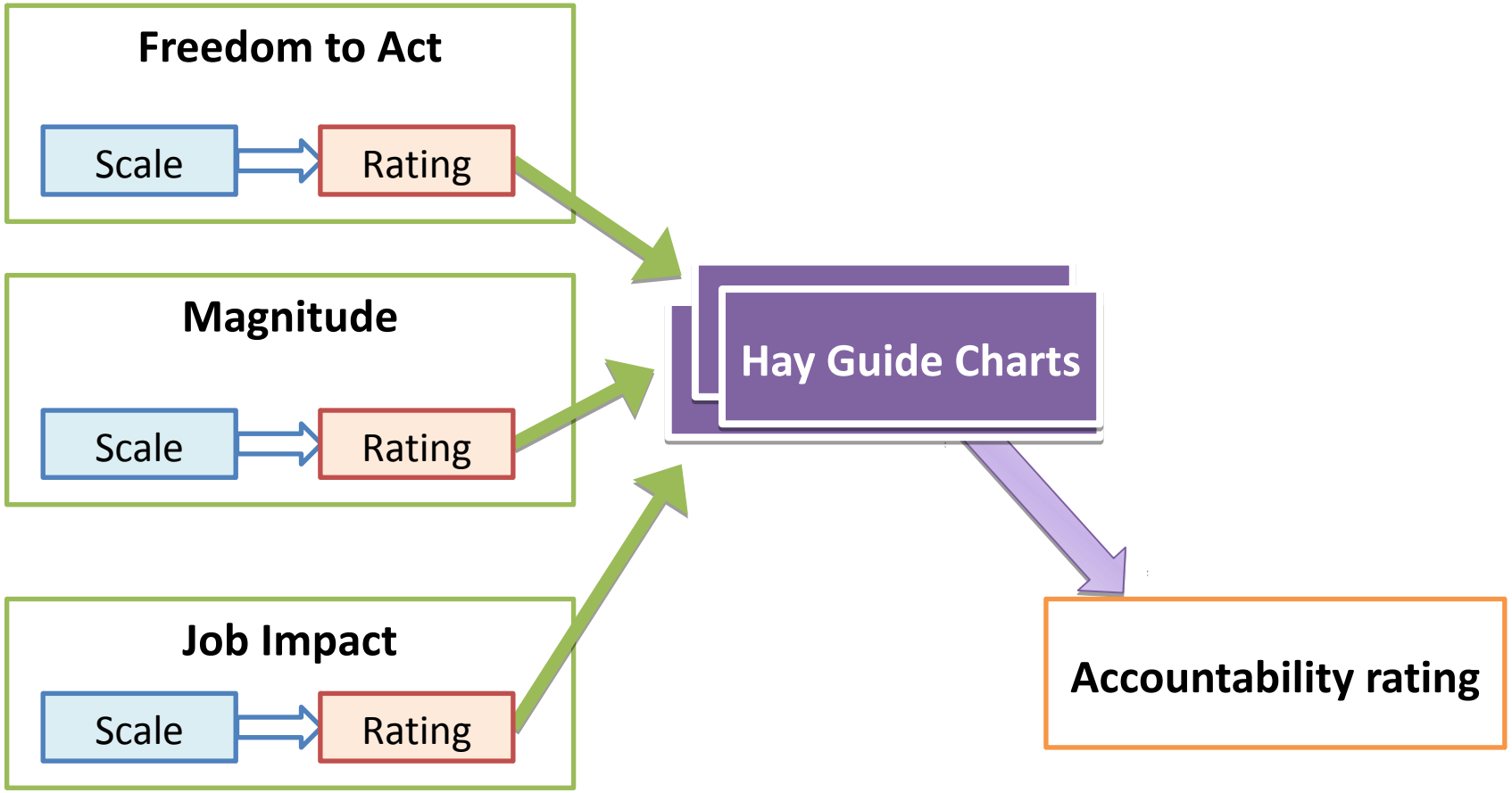
Hay method


Job-Specific Knowledge
Scale
Rating
Managerial Breadth
Scale
Rating
Human Relations Skills
Scale
Rating
Hay Guide Charts
Know-How rating

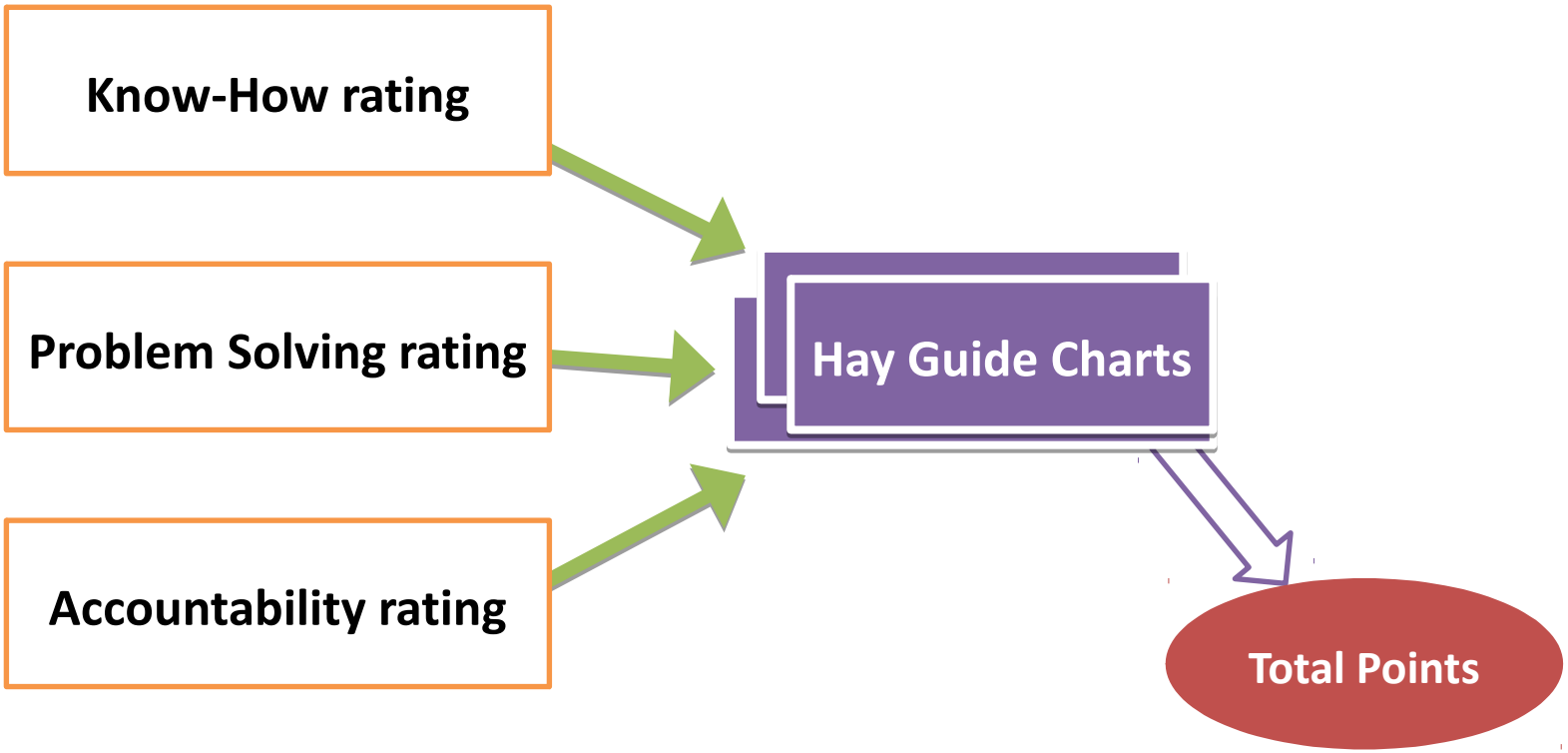
Hay method


Thinking Environment
Scale
Rating
Thinking Challenge
Scale
Rating
Hay Guide Charts
Problem Solving rating

Hay method

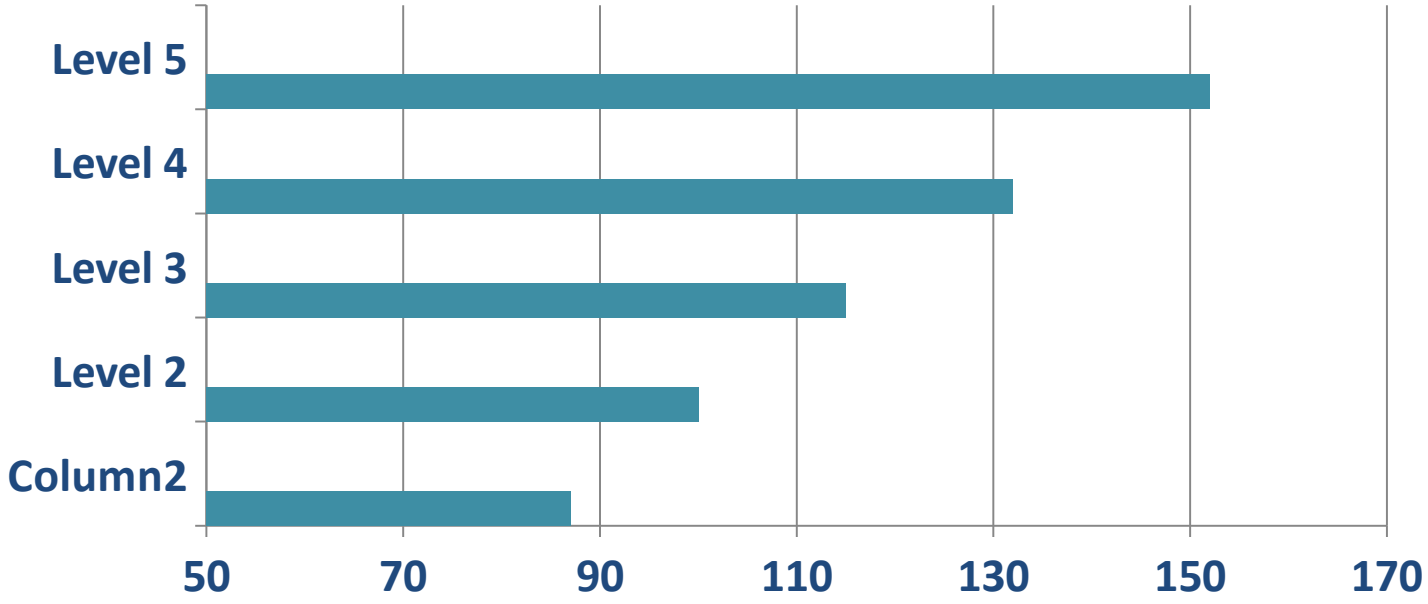

Freedom to Act
Scale
Rating
Magnitude
Scale
Rating
Job Impact
Scale
Rating
Hay Guide Charts
Accountability rating

Hay method


Know-How rating
Problem Solving rating
Accountability rating
Hay Guide Charts
Total Points

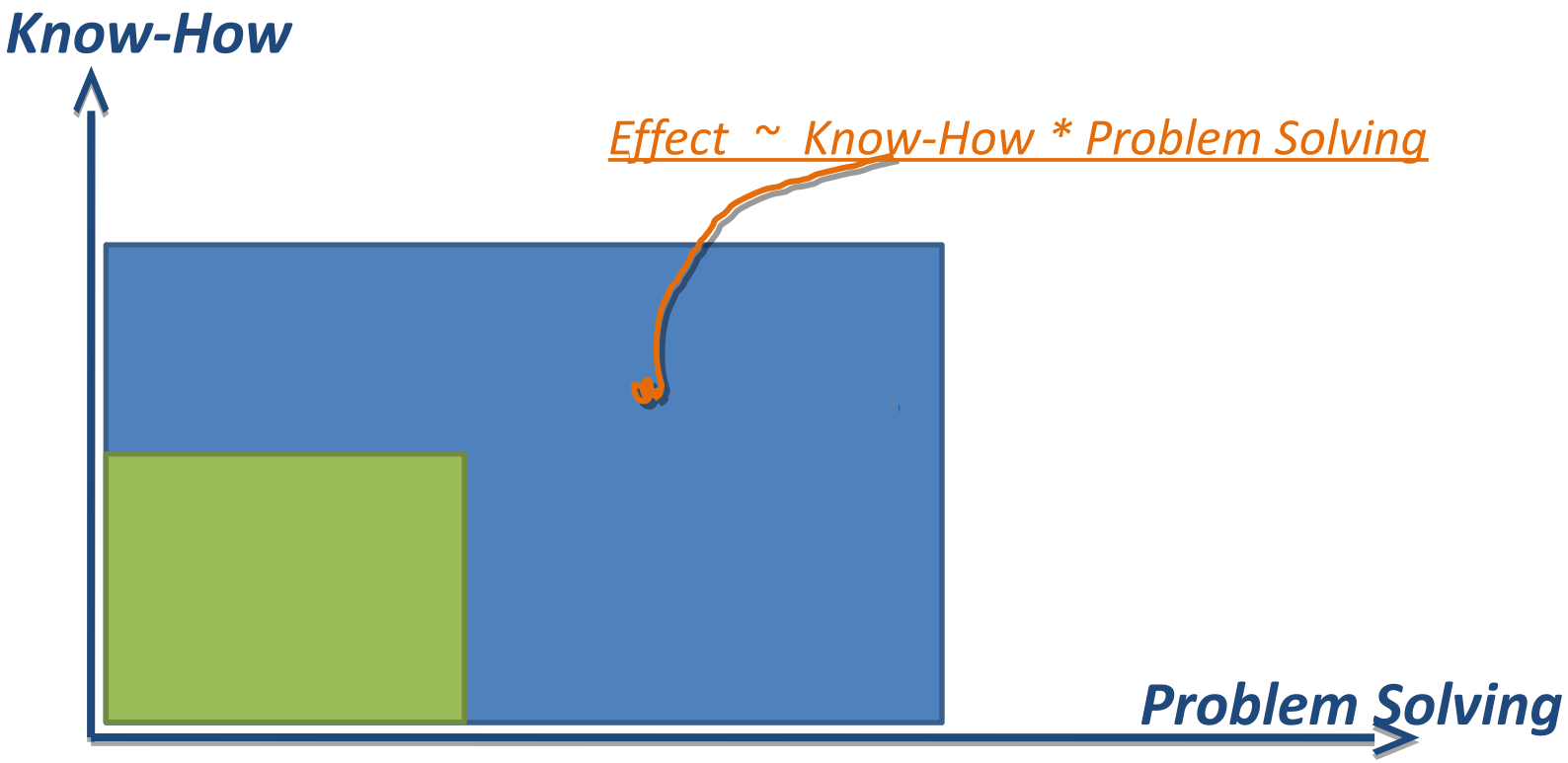
Hay method

Noticeable difference between scale levels is 15%



Hay method

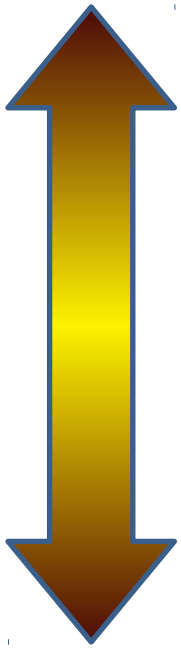
Problem Solving is a percentage of Know-How: “you think with what you know”



Hay method

Accountability and Problem Solving ratings are related to the job profile

Accountability



Problem Solving

- Coordination
- Regulation
- Process
- Consulting
- Analysis
- Applied Research
- Original Research

Developers evaluation system

Developer's job profile specifics

- Biased towards problem solving
- Wide range of knowledge areas and skills
- Invariable factors:
  - Thinking Environment
  - Magnitude
  - Job Impact

Developers evaluation system

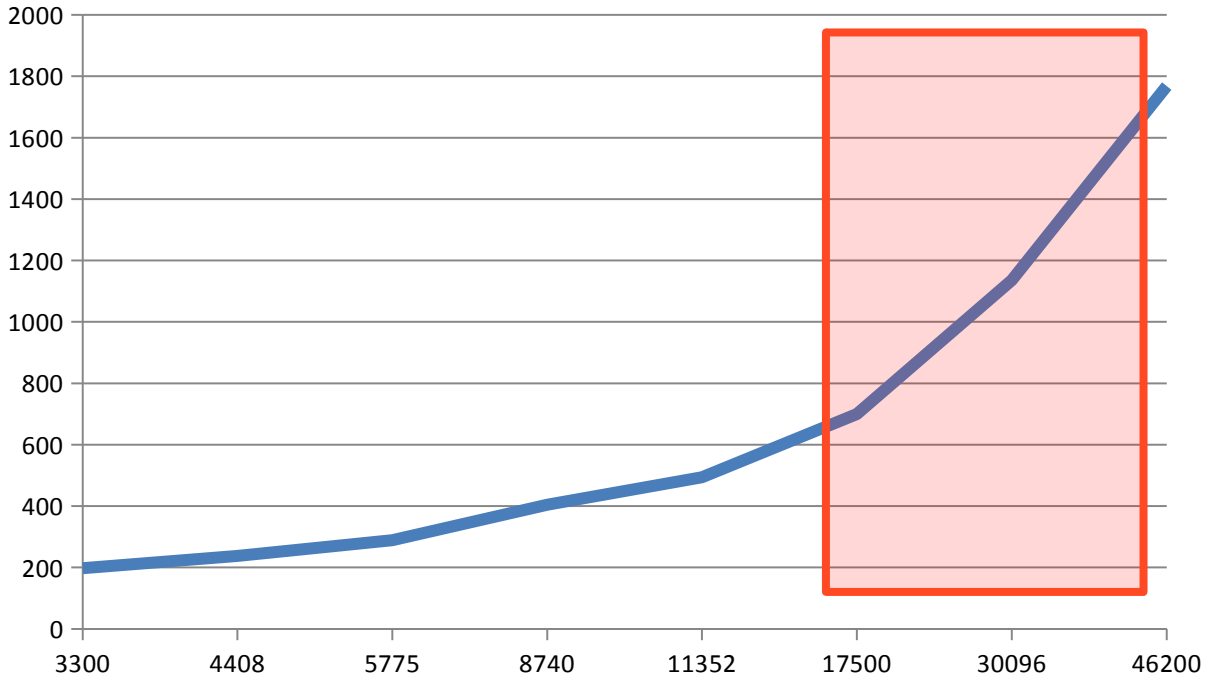
Hay Guide Charts: area with high Know-How and Problem Solving ratings

PS	Short	Know-How Points													
%	Profile	115	132	152	175	200	230	264	304	350	400	460	528	608	700
25	=	173	**198**	**228**	261										
25	+1	177	**203**	**233**	268										
29	-1	177	203	233	**268**	307	353								
29	=	181	**208**	**238**	**275**	**314**	362								
29	+1	186	213	**245**	**282**	**323**	372								
33	-1	186	213	245	**282**	**323**	**372**								
33	=	191	218	**252**	**289**	**332**	**382**	438							
29	+2	191	220	252	**291**	**333**	383	440							
33	+1				**298**	**342**	**393**	**451**							
38	-1				298	342	393	451	519	597					
38	=				307	352	**404**	464	**534**	614					
33	+2				**308**	**353**	**406**	**466**	536	617					
43	-2					353	406	466	536	617	707	812			
38	+1					363	**417**	479	**551**	**634**	727	835			
43	-1						**417**	**479**	551	634	727	835			
43	=						430	**494**	**568**	**654**	**750**	860			
38	+2						**432**	496	**571**	**657**	[illegible]	[illegible]			
50	-2						432	496	571	657	752	865	992	1142	
43	+1							**511**	**588**	**677**	**775**	890	1022	1176	
50	-1							511	588	677	775	890	1022	1176	
50	=							528	608	**700**	**800**	**920**	1056	1216	
43	+2							**531**	**611**	**702**	**805**	**924**	162	1222	
57	-2								611	702	805	924	1062	1222	
50	+1									**725**	**830**	**954**	**1096**	1262	1450
57	-1									725	830	954	1096	1262	1450
57	=									750	860	988	**1136**	**1308**	1500
50	+2									**755**	**864**	**994**	**1142**	**1312**	1510
66	-2										864	994	1142	1312	1510
57	+1										894	**1028**	**1182**	**1358**	1560
57	+2										934	**1074**	1232	**1418**	**1628**
66	+2														**1768**

Source: http://www.beta.mmb.state.mn.us/doc/comp/hay/hay-manual.pdf

Developers evaluation system

Total Points = *f* (Know How * Problem Solving) *



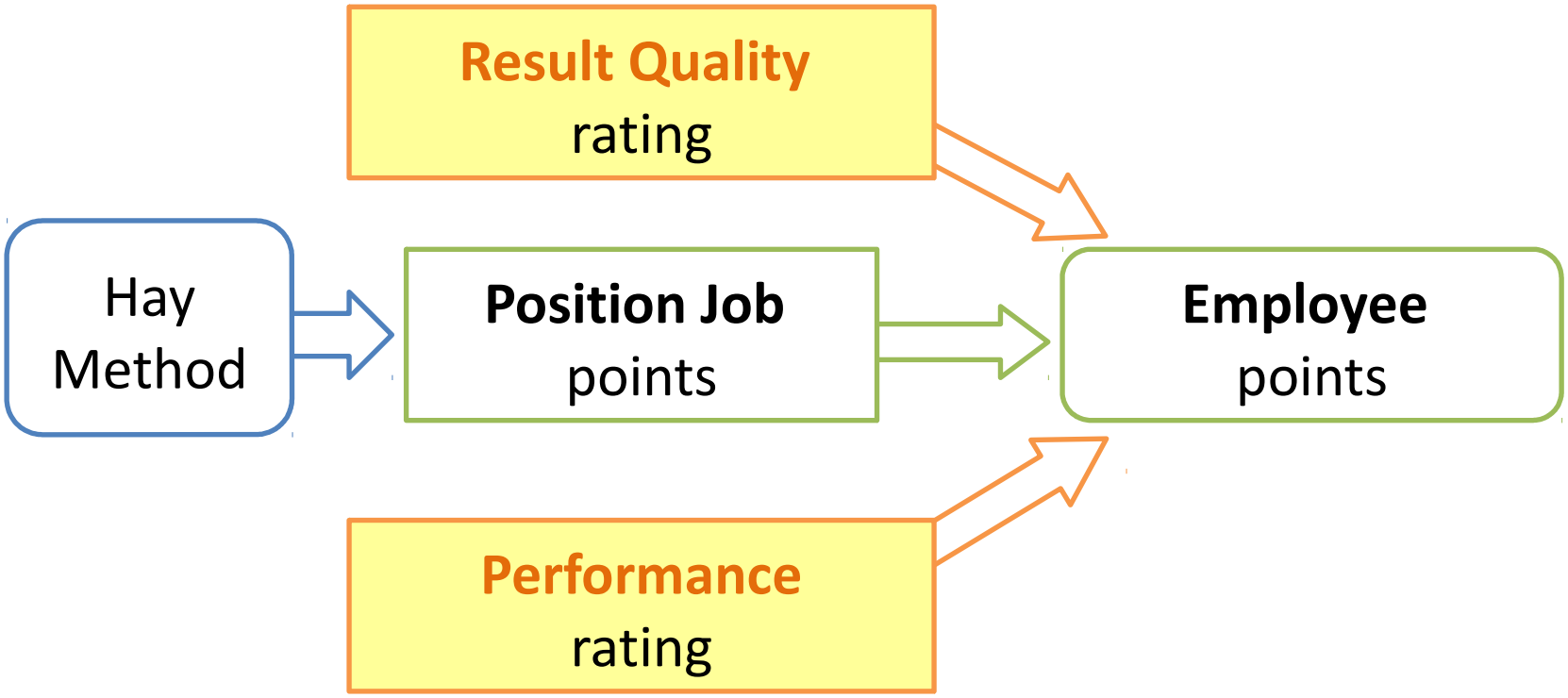
Almost linear part of the chart

**Based on Hay Guide Charts numbers*

Developers evaluation system

Developers evaluation system

Introducing additional factors

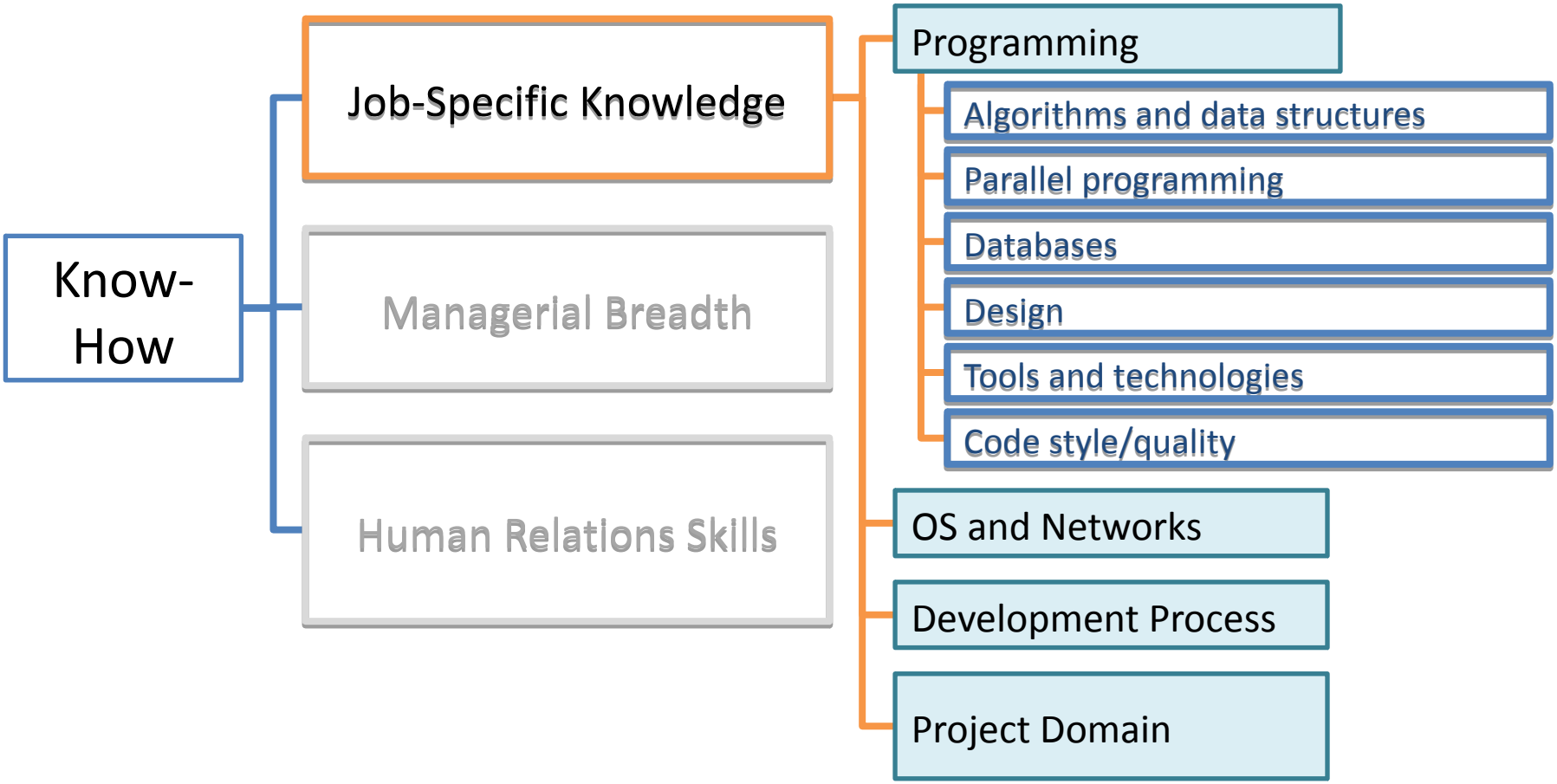


Solution

1. Decompose evaluation factors

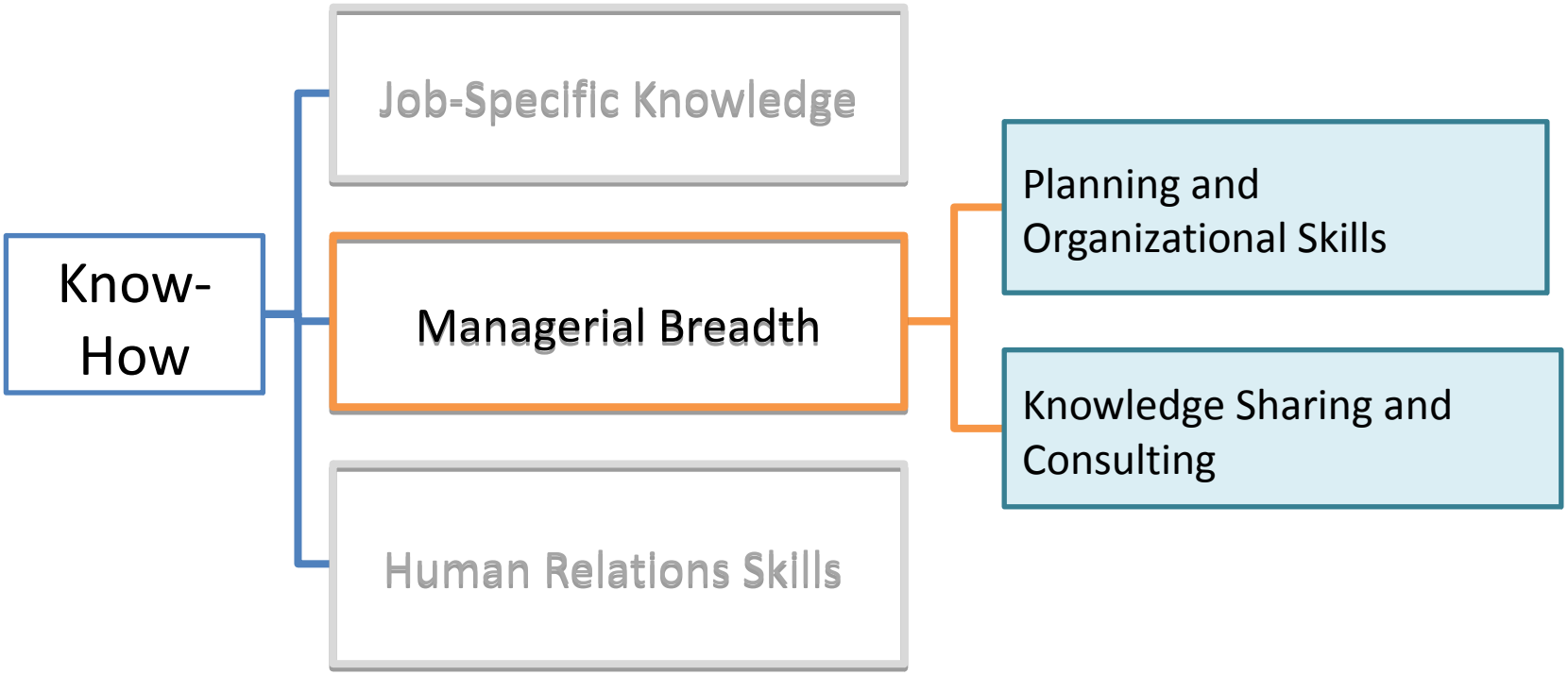
Solution

Decompose evaluation factors



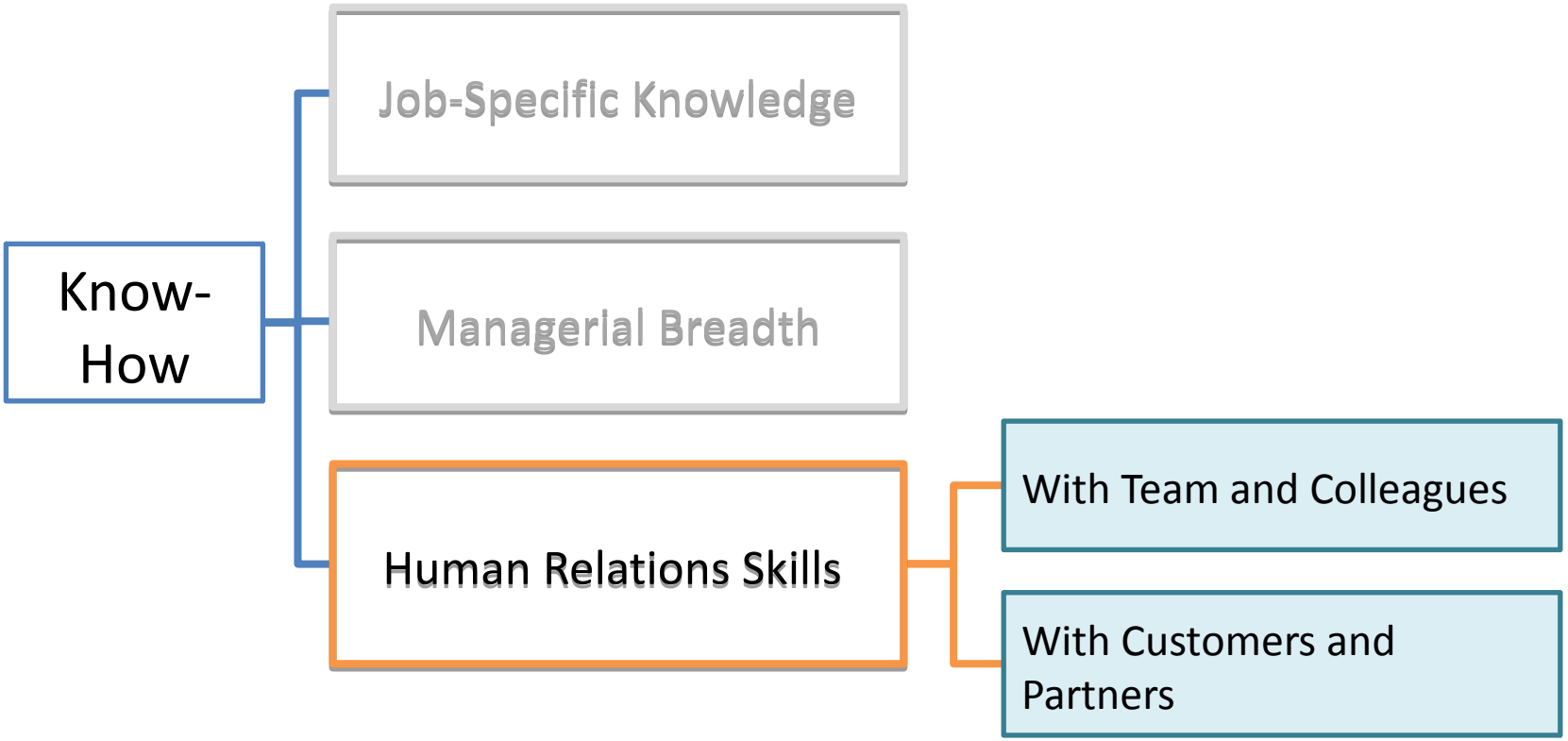
Solution

Decompose evaluation factors



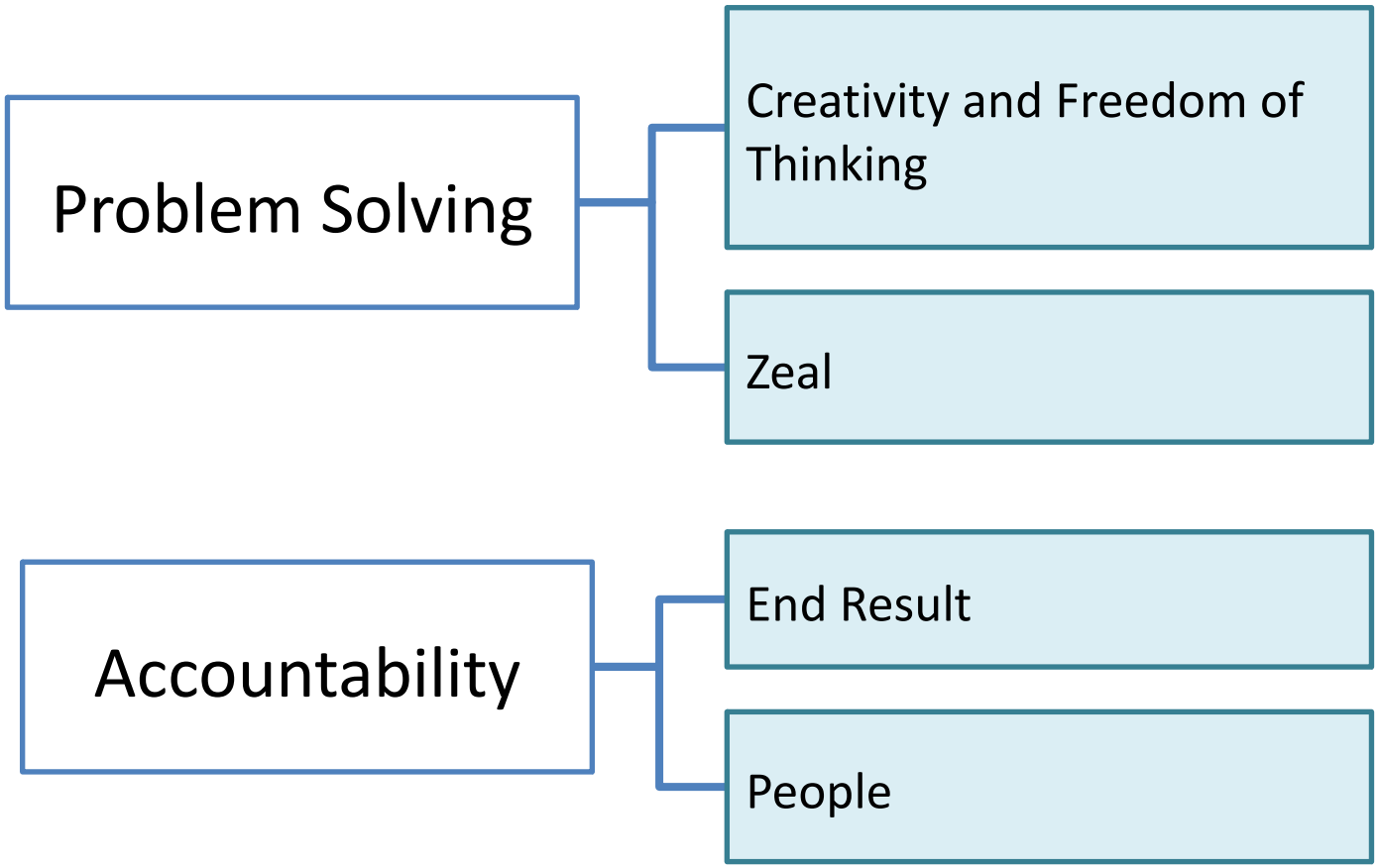
Solution

Decompose evaluation factors



Solution

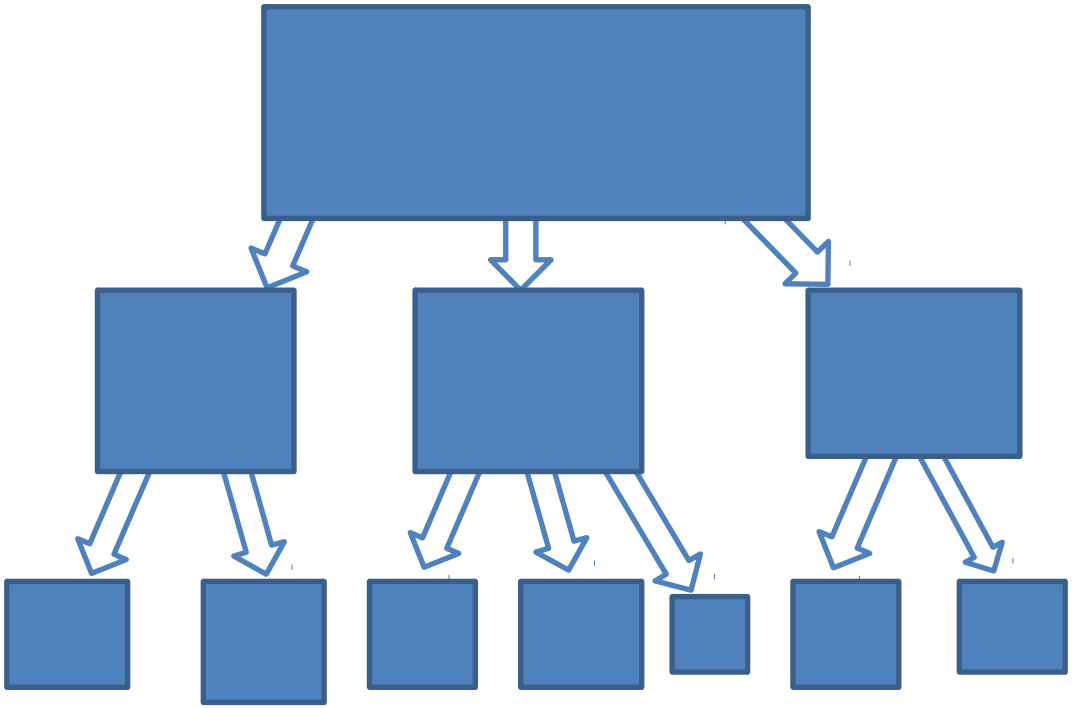
Decompose evaluation factors

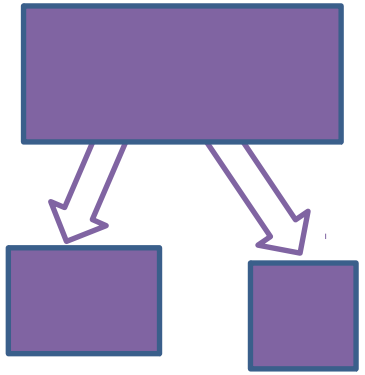


Solution

Decompose evaluation factors

Factor decomposition depth depends on factor significance





Solution

1. Decompose evaluation factors
2. **Define a scale and level descriptions for each subfactor**

Solution

Subfactor scale example

Know-How > Job-specific knowledge > Programming > Parallel Programming

Level	Description
1	Knows about “multithreading” but can write only single-threaded code.
2	Understands concurrent resource access problems. Knows how deadlock appears and how to avoid it in simplest case.
3	Familiar with the concept of volatile and atomic variables, can apply them appropriately. Knows thread-safe structures design, thread starting/stopping and synchronizing procedures. Can implement a thread pool, develop code accessing a set of resources in multithreaded environment, etc.
4	Understands performance problems in multithreaded environment and the ways to prevent them. Understands synchronization primitives, can deal with them (read/write locks, reentrant locks, etc). Can deal with concurrent data structures. Familiar with non-blocking and lock-free algorithms.

Solution

Subfactor scale example

Know-How > Managerial Breadth > Planning and organizational skills

Level	Description
1	Fulfils without assistance only simple tasks if expound in detail. Needs mentoring on permanent basis.
2	Copes with planning and fulfilling well-defined and prioritized tasks of average size with a number of stages/subtasks. Identifies gaps and contradictions in task definition, requests explanation. Needs consulting assistance from time to time. Effort estimates may be a few times higher/lower than real one.
3	Works out details, plans and fulfils without any assistance complex tasks with high degree of uncertainty. Proposes solutions in case of missing requirements, approaches to resolution of technical issues. Gives accurate enough estimates of efforts (20%-50% error).
4	Skilled enough to coordinate working activities of group of developers on common task, including requrements elaboration, breaking down onto stages/subtasks, resource planning, task assignment, control over the progress, etc. If needed, initiates discussions and research activities as part of the task execution. In addition to development coordinates appropriate update of requirements, documentation and other artifacts related to product development. Provides reliable estimates on time/resources needed for the entire scope of work with 20%-50% precision.

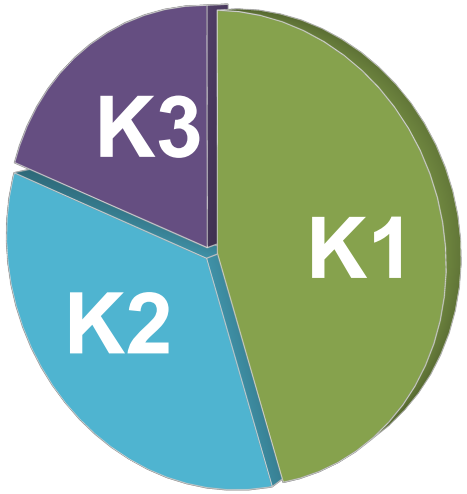
Solution

1. Decompose evaluation factors
2. Define a scale and level descriptions for each subfactor
3. **Define weight coefficients of subfactors**

Solution

Define weight coefficients of subfactors

[Know-How] = **K1** * [Job-Specific Knowledge] +
K2 * [Managerial Breadth] +
K3 * [Human Relations Skills]



$\Sigma Ki = 1$

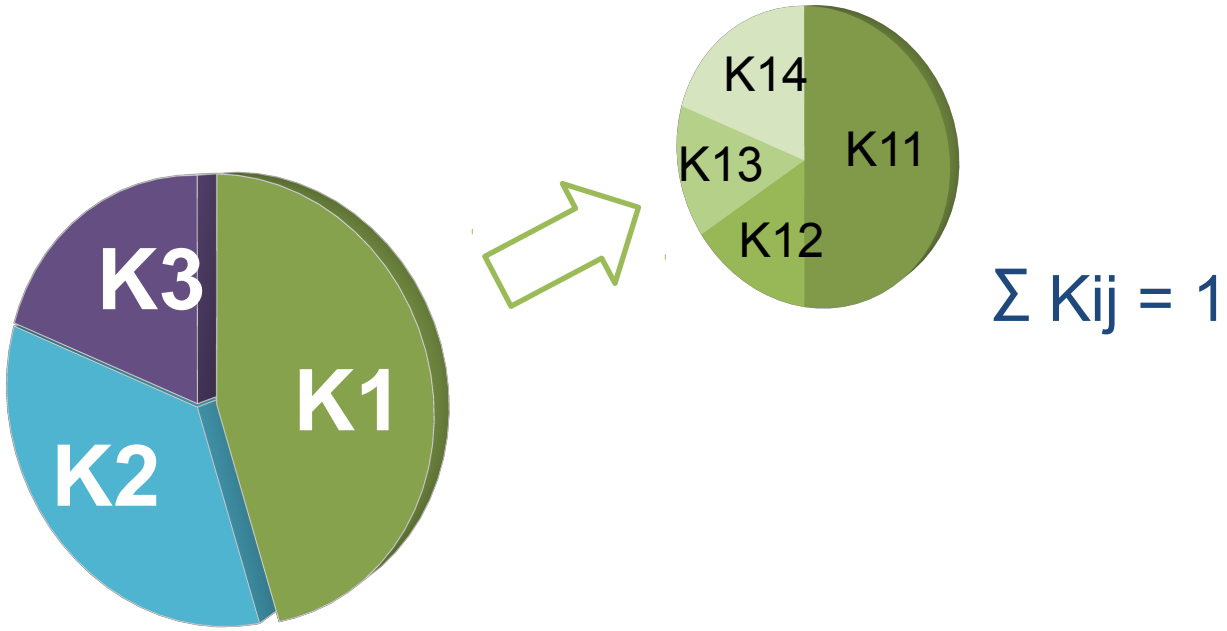
Solution

Define weight coefficients of subfactors

[Job-Specific Knowledge] =

K11 * *[Programming]* + **K12** * *[OS and Networks]* +

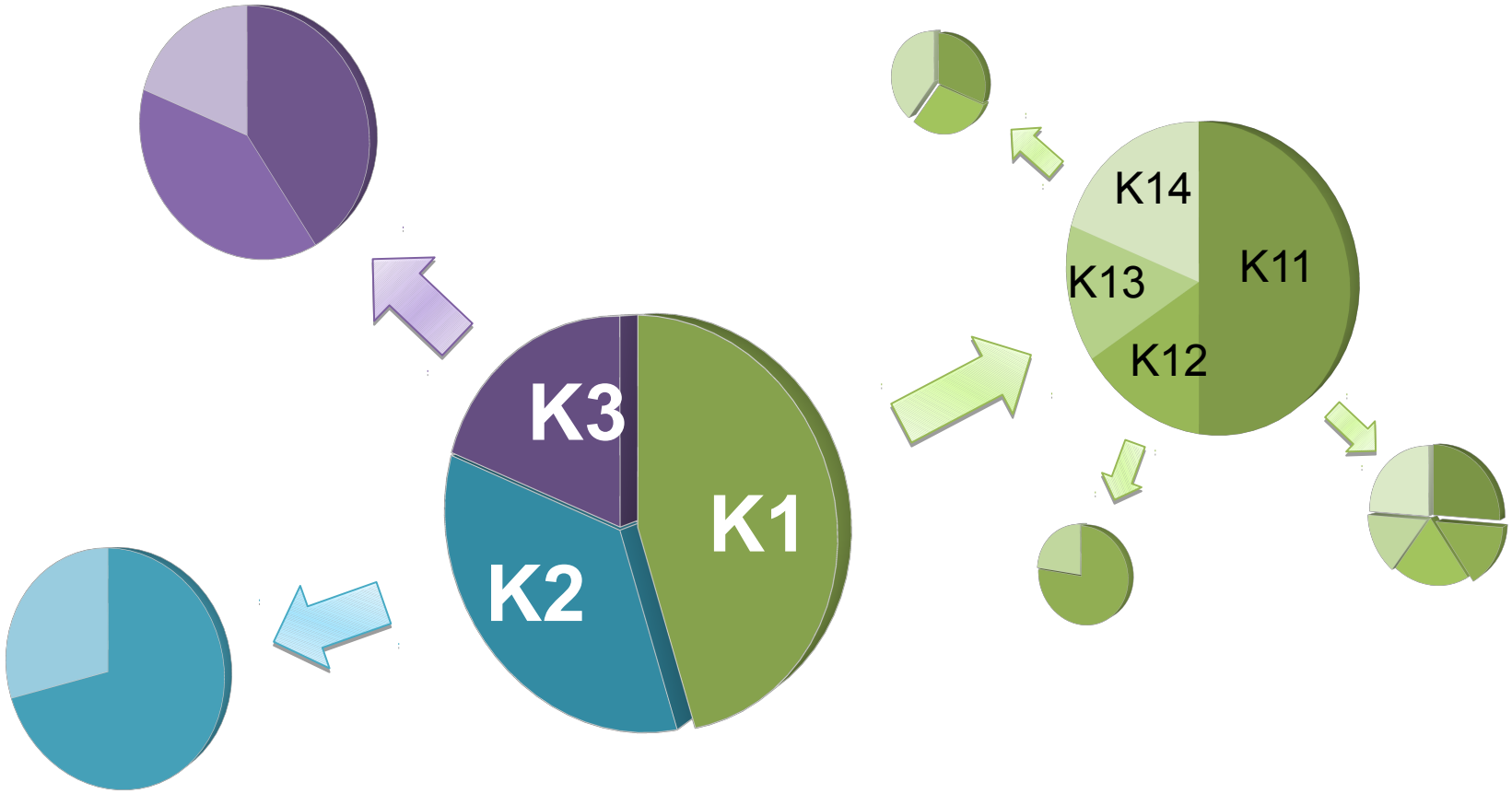
K13 * *[Development Process]* + **K14** * *[Project Domain]* + …



$\Sigma K_{ij} = 1$

Solution

Define weight coefficients of subfactors



Solution

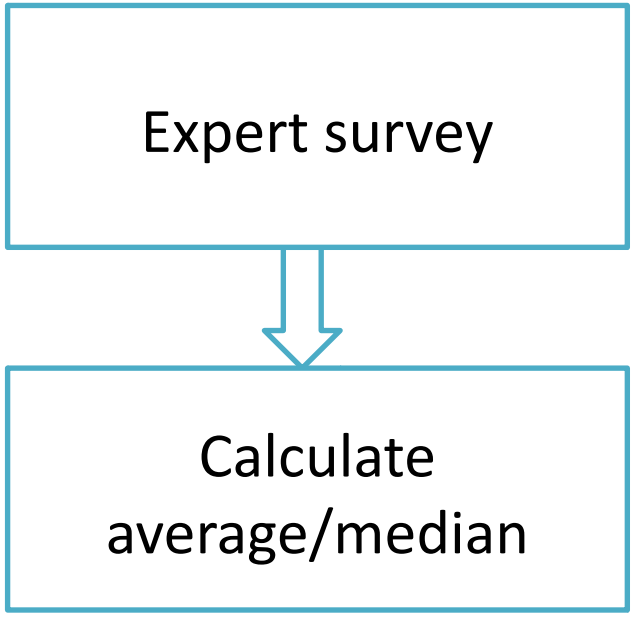
Define weight coefficients of subfactors

[Problem Solving] = **N1** * *[Creativity]* + **N2** * *[Zeal]* + …

[Accountability] = **M1** * *[End Result]* + **M2** * *[People]* + …

Solution

Define weight coefficients of subfactors



Solution

1. Decompose evaluation factors
2. Define a scale and level descriptions for each subfactor
3. Define weight coefficients of subfactors
4. **Define a formula for Total Points calculation**

Solution

Define a formula for Total Points calculation

Base rating:

Know-How * Problem Solving

Solution

Define a formula for Total Points calculation

Take **accountability** into account

Know-How * Problem Solving*

(1 + **K1** * **Accountability**)

Solution

Define a formula for Total Points calculation

With respect to **effort**:

Know-How * Problem Solving *

(1 + **K1** * Accountability) *

(**K2** * **Performance** + **K3** * **Quality**)

Solution

Define a formula for Total Points calculation

Total Points = Know-How * Problem Solving *
(1 + **K1** * Accountability) *
(**K2** * Performance + **K3** * Quality)

Solution

1. Decompose evaluation factors
2. Define a scale and level descriptions for each subfactor
3. Define weight coefficients of subfactors
4. Define a formula for Total Points calculation
5. **Fine-tune parameters**

Solution

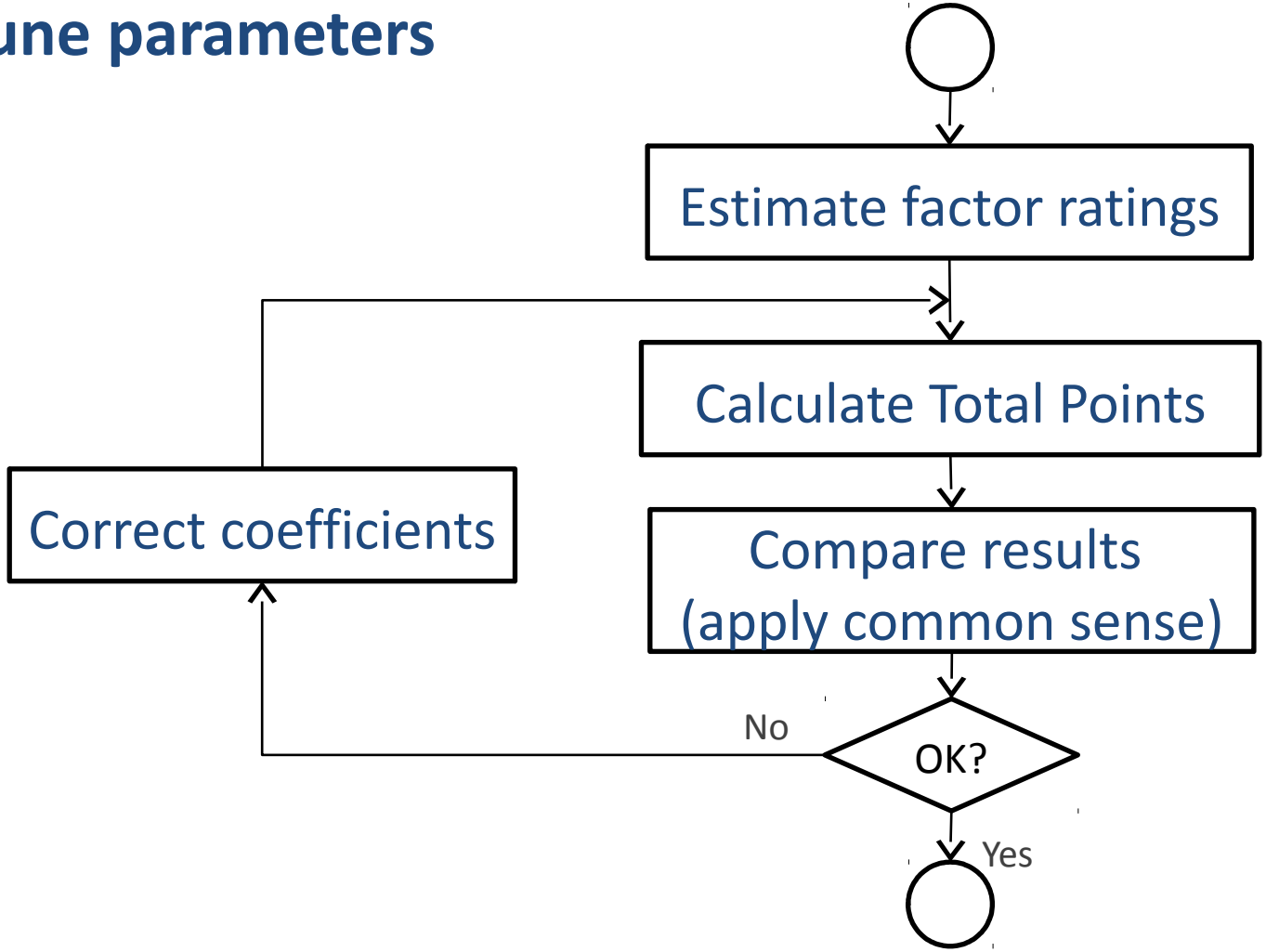
Fine-tune parameters

Using synthetic profiles, e.g.

-  Student/Probationer
-  «Working Horse»
-  Experienced Senior Developer
-  Lead Developer, Architect
-  Expert-Consultant

Solution

Fine-tune parameters



Solution

1. Decompose evaluation factors
2. Define a scale and level descriptions for each subfactor
3. Define weight coefficients of subfactors
4. Define a formula for Total Points calculation
5. Fine-tune parameters
6. **Define grades**

Solution

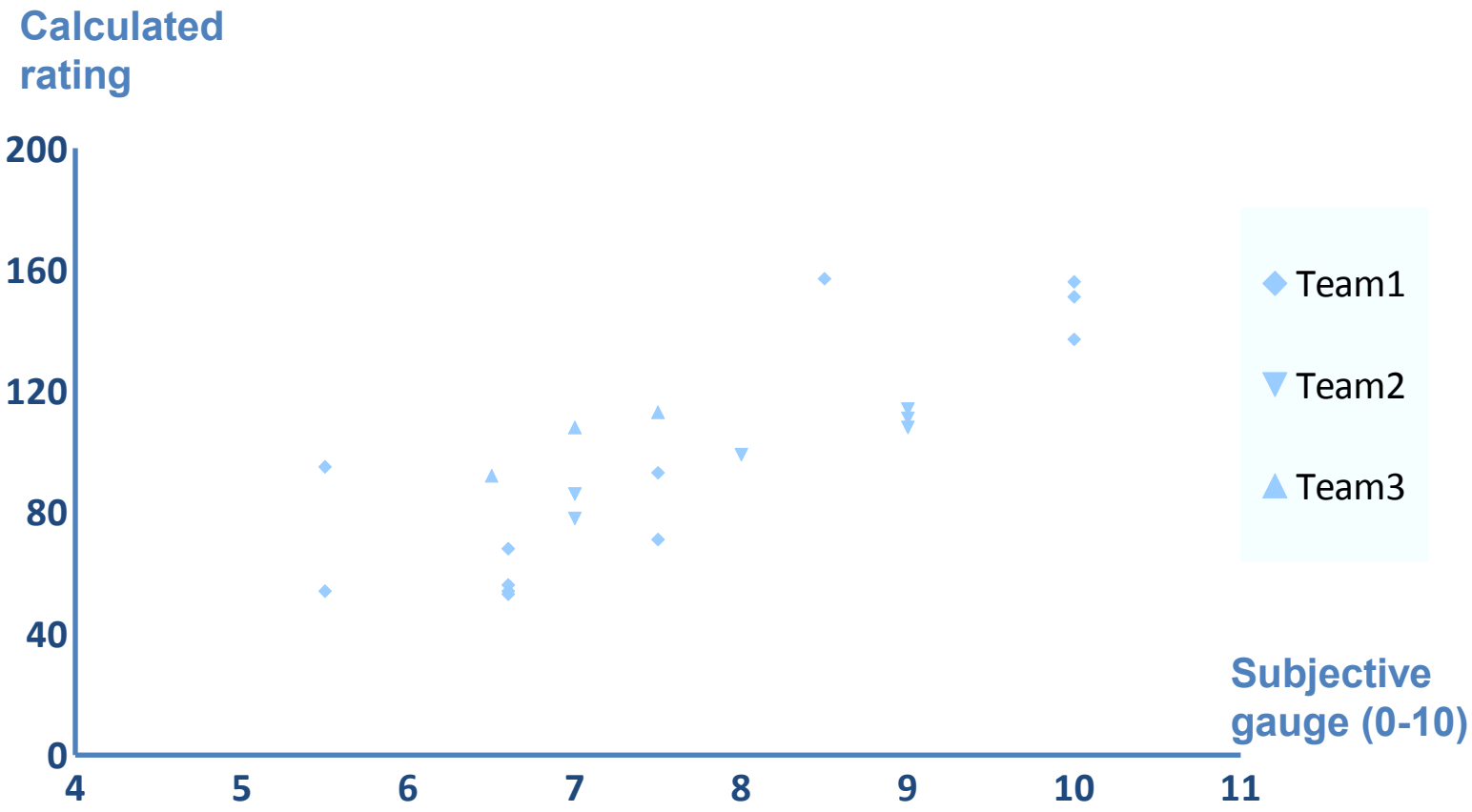
Define grades

Total points range	Grade
50 - 70	D1: Probationer
65 - 90	D2: Junior Developer
85 - 110	D3: Developer
…	…
200+	D7: Expert

Application Experience

Correlation between calculated Total Points and «gut feeling» based rating depends on experience and objectivity of manager/rater

Application Experience

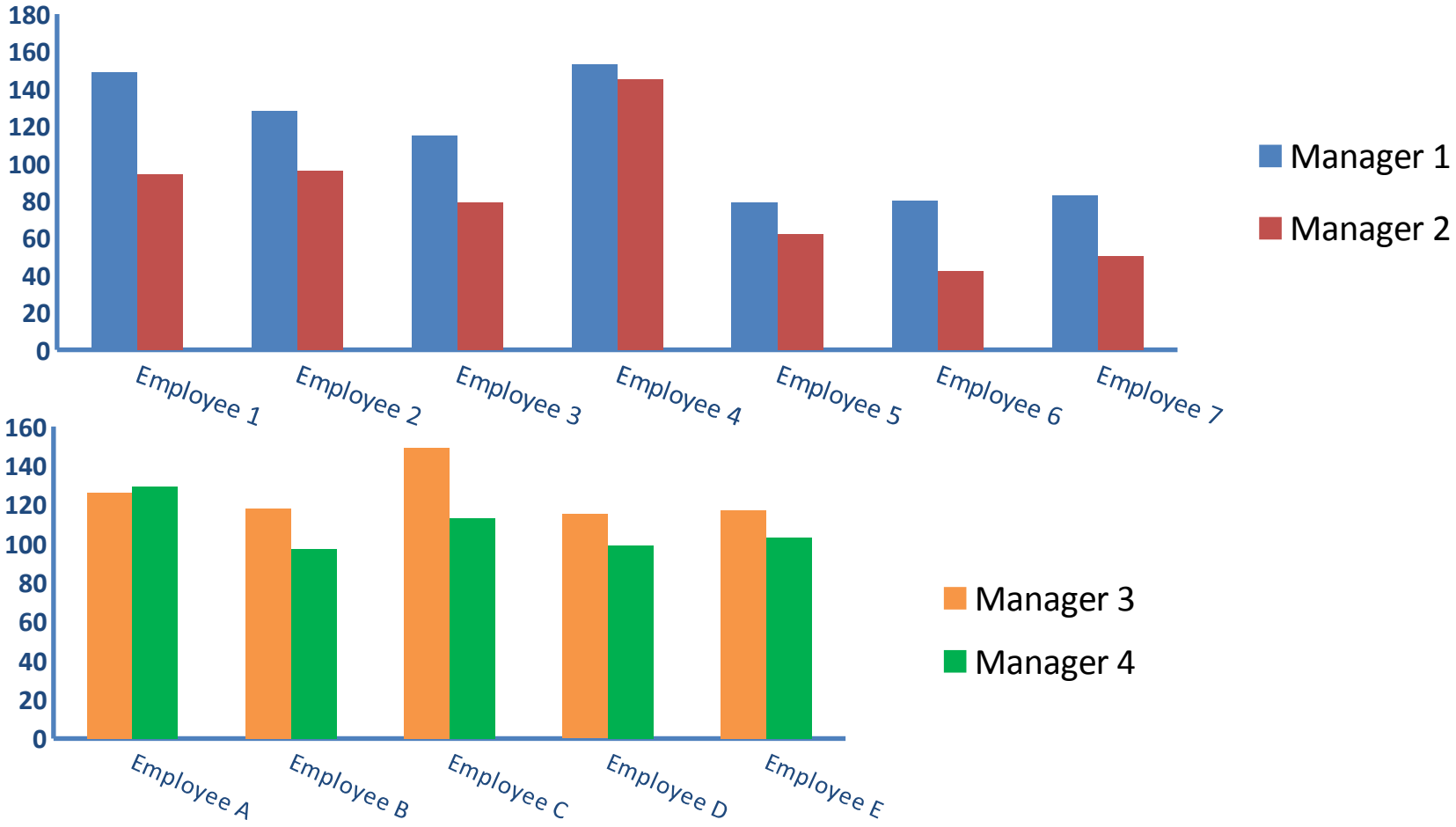


Calculated Total Points vs subjective gauge – based on **real** data

Application Experience

Each particular manager rates on his specific level of tolerance

Application Experience



Comparison of gauges made by 2 managers independently – based on **real** data

Application Experience

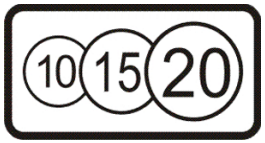
- Calculated rating has high enough correlation with subjective gauge
- Evaluation results become much more transparent
- Specific ratings may be confirmed by artifacts
- It is hard for some managers to estimate certain knowledge factors without assistance

Conclusions

- You also can do this
  - with respect to your company's needs
  - applicable not only to developers

Conclusions

- **You do the evaluation anyway**



  - **but you can do it better**

References

1. Hay method official white paper: http://www.haygroup.com/downloads/au/Guide_Chart-Profile_Method_of_Job_Evaluation_Brochure_web.pdf
2. Hay method application manual by State of Minnesota, USA: http://www.beta.mmb.state.mn.us/doc/comp/hay/hay-manual.pdf
3. Hay method application by Alberta province, Canada: http://www.chr.alberta.ca/learning/competencies/apsmodel/aps-competency-model.pdf
4. Mercer's IPE: http://www.ovc.lt/uploads/File/Johan_Ericsson_presentation.pdf
5. Discussion on developers efficiency evaluation: http://habrahabr.ru/post/101906/
6. Programmer competency matrix: http://sijinjoseph.com/programmer-competency-matrix/
7. 360 degree feedback: http://en.wikipedia.org/wiki/360-degree_feedback

Thanks!

Questions?

Valentin Anoprenko

anoprenko@devexperts.com