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

<table>
<thead>
<tr>
<th></th>
<th>2n (Level 0)</th>
<th>n² (Level 1)</th>
<th>n (Level 2)</th>
<th>log(n) (Level 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computer Science</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data structures</td>
<td>Doesn’t know the difference between Array and LinkedList</td>
<td>Able to explain and use Arrays, LinkedLists, Dictionaries etc in practical programming tasks</td>
<td>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.</td>
<td>Knowledge of advanced data structures like B-trees, binomial and fibonaccic heaps, AVL/Red Black trees, Splay Trees, Skip Lists, tries etc.</td>
</tr>
<tr>
<td>algorithms</td>
<td>Unable to find the average of numbers in an array (It’s hard to believe but I’ve interviewed such candidates)</td>
<td>Basic sorting, searching and data structure traversal and retrieval algorithms</td>
<td>Tree, Graph, simple greedy and divide and conquer algorithms, is able to understand the relevance of the levels of this matrix.</td>
<td>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.</td>
</tr>
<tr>
<td>systems programming</td>
<td>Doesn’t know what a compiler, linker or interpreter is</td>
<td>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.</td>
<td>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.</td>
<td>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...</td>
</tr>
<tr>
<td><strong>Software Engineering</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>source code version control</td>
<td>Folder backups by date</td>
<td>VSS and beginning CVS/SVN user</td>
<td>Proficient in using CVS and SVN features. Knows how to branch and merge, use patches setup repository properties etc.</td>
<td>Knowledge of distributed VCS systems. Has tried out Bzr/Mercurial/Darcs/Git</td>
</tr>
<tr>
<td>build automation</td>
<td>Only knows how to build from IDE</td>
<td>Knows how to build the system from the command line</td>
<td>Can setup a script to build the basic system</td>
<td>Can setup a script to build the system and also documentation, installers, generate release notes and tag the code in source control</td>
</tr>
<tr>
<td>automated testing</td>
<td>Thinks that all testing is the job of the tester</td>
<td>Has written automated unit tests and comes up with good unit test cases for the code that is being written</td>
<td>Has written code in TDD manner</td>
<td>Understands and is able to setup automated functional, load/performance and UI tests</td>
</tr>
<tr>
<td>Programming</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>problem decomposition</td>
<td>Only straight line code with copy paste for reuse</td>
<td>Able to break up problem into multiple functions</td>
<td>Able to come up with reusable functions/objects that solve the overall problem</td>
<td>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.</td>
</tr>
</tbody>
</table>
Possible approaches

«360° 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

Know-How rating
Hay method

Thinking Environment

Thinking Challenge

Hay Guide Charts

Scale ➔ Rating

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”

\[
\text{Effect} \sim \text{Know-How} \times \text{Problem Solving}
\]
Hay method

Accountability and Problem Solving ratings are related to the job profile
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

<table>
<thead>
<tr>
<th>PS</th>
<th>Short</th>
<th>Profile</th>
<th>115</th>
<th>132</th>
<th>152</th>
<th>175</th>
<th>200</th>
<th>230</th>
<th>264</th>
<th>304</th>
<th>350</th>
<th>400</th>
<th>460</th>
<th>528</th>
<th>608</th>
<th>700</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
<td>173</td>
<td>198</td>
<td>228</td>
<td>261</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>+1</td>
<td>177</td>
<td>203</td>
<td>233</td>
<td>268</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>-1</td>
<td>177</td>
<td>203</td>
<td>233</td>
<td>268</td>
<td>307</td>
<td>353</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>=</td>
<td>181</td>
<td>208</td>
<td>238</td>
<td>275</td>
<td>314</td>
<td>362</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>+1</td>
<td>186</td>
<td>213</td>
<td>245</td>
<td>282</td>
<td>323</td>
<td>372</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>-1</td>
<td>186</td>
<td>213</td>
<td>245</td>
<td>282</td>
<td>323</td>
<td>372</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>=</td>
<td>181</td>
<td>218</td>
<td>252</td>
<td>289</td>
<td>332</td>
<td>382</td>
<td>438</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>+2</td>
<td>191</td>
<td>220</td>
<td>252</td>
<td>291</td>
<td>333</td>
<td>383</td>
<td>440</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>-1</td>
<td>298</td>
<td>342</td>
<td>393</td>
<td>451</td>
<td>519</td>
<td>577</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>=</td>
<td>307</td>
<td>352</td>
<td>404</td>
<td>464</td>
<td>534</td>
<td>614</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>+2</td>
<td>308</td>
<td>353</td>
<td>406</td>
<td>466</td>
<td>538</td>
<td>617</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>-2</td>
<td>363</td>
<td>417</td>
<td>479</td>
<td>551</td>
<td>634</td>
<td>727</td>
<td>835</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>+1</td>
<td>363</td>
<td>417</td>
<td>479</td>
<td>551</td>
<td>634</td>
<td>727</td>
<td>835</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>=</td>
<td>363</td>
<td>417</td>
<td>479</td>
<td>551</td>
<td>634</td>
<td>727</td>
<td>835</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>+2</td>
<td>363</td>
<td>417</td>
<td>479</td>
<td>551</td>
<td>634</td>
<td>727</td>
<td>835</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>-2</td>
<td>432</td>
<td>496</td>
<td>571</td>
<td>657</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>+1</td>
<td>511</td>
<td>588</td>
<td>677</td>
<td>775</td>
<td>890</td>
<td>1022</td>
<td>1176</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>=</td>
<td>511</td>
<td>588</td>
<td>677</td>
<td>775</td>
<td>890</td>
<td>1022</td>
<td>1176</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>+2</td>
<td>511</td>
<td>588</td>
<td>677</td>
<td>775</td>
<td>890</td>
<td>1022</td>
<td>1176</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>-2</td>
<td>611</td>
<td>702</td>
<td>805</td>
<td>924</td>
<td>1062</td>
<td>1222</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>=</td>
<td>611</td>
<td>702</td>
<td>805</td>
<td>924</td>
<td>1062</td>
<td>1222</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>+2</td>
<td>611</td>
<td>702</td>
<td>805</td>
<td>924</td>
<td>1062</td>
<td>1222</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>+1</td>
<td>755</td>
<td>864</td>
<td>994</td>
<td>1142</td>
<td>1312</td>
<td>1510</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>-1</td>
<td>755</td>
<td>864</td>
<td>994</td>
<td>1142</td>
<td>1312</td>
<td>1510</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>=</td>
<td>755</td>
<td>864</td>
<td>994</td>
<td>1142</td>
<td>1312</td>
<td>1510</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>+2</td>
<td>755</td>
<td>864</td>
<td>994</td>
<td>1142</td>
<td>1312</td>
<td>1510</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>-2</td>
<td>894</td>
<td>1028</td>
<td>1182</td>
<td>1358</td>
<td>1560</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>+1</td>
<td>894</td>
<td>1028</td>
<td>1182</td>
<td>1358</td>
<td>1560</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>=</td>
<td>894</td>
<td>1028</td>
<td>1182</td>
<td>1358</td>
<td>1560</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>+2</td>
<td>894</td>
<td>1028</td>
<td>1182</td>
<td>1358</td>
<td>1560</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Developers evaluation system

Total Points = \( f (\text{Know How} \times \text{Problem Solving}) \)

*Based on Hay Guide Charts numbers*
Developers evaluation system

Hay Method → Position points → Employee points
Developers evaluation system

Introducing additional factors

Hay Method

Result Quality rating

Position Job points

Performance rating

Employee points
Solution

1. Decompose evaluation factors
Solution

Decompose evaluation factors

Know-How

Job-Specific Knowledge

- Managerial Breadth
- Human Relations Skills

Programming

- Algorithms and data structures
- Parallel programming
- Databases
- Design
- Tools and technologies
- Code style/quality

OS and Networks

Development Process

Project Domain
Solution

Decompose evaluation factors

- **Know-How**
  - Job-Specific Knowledge
  - Managerial Breadth
  - Human Relations Skills
- Planning and Organizational Skills
- Knowledge Sharing and Consulting
Solution

Decompose evaluation factors

Know-How

Job-Specific Knowledge

Managerial Breadth

Human Relations Skills

With Team and Colleagues

With Customers and Partners
Solution

Decompose evaluation factors

Problem Solving
  - Creativity and Freedom of Thinking
  - Zeal

Accountability
  - End Result
  - People
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**

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Knows about “multithreading” but can write only single-threaded code.</td>
</tr>
<tr>
<td>2</td>
<td>Understands concurrent resource access problems. Knows how deadlock appears and how to avoid it in simplest case.</td>
</tr>
<tr>
<td>3</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>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.</td>
</tr>
</tbody>
</table>
## Solution

### Subfactor scale example

**Know-How > Managerial Breadth > Planning and organizational skills**

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fulfils without assistance only simple tasks if expound in detail. Needs mentoring on permanent basis.</td>
</tr>
<tr>
<td>2</td>
<td>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.</td>
</tr>
<tr>
<td>3</td>
<td>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).</td>
</tr>
<tr>
<td>4</td>
<td>Skilled enough to coordinate working activities of group of developers on common task, including requirements 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.</td>
</tr>
</tbody>
</table>
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

\[ [\text{Know-How}] = K_1 \times [\text{Job-Specific Knowledge}] + K_2 \times [\text{Managerial Breadth}] + K_3 \times [\text{Human Relations Skills}] \]

\[ \sum K_i = 1 \]
Solution

Define weight coefficients of subfactors

\[
\text{[Job-Specific Knowledge]} = \sum K_{ij} = 1
\]

\[
K_{11} \times [\text{Programming}] + K_{12} \times [\text{OS and Networks}] +
K_{13} \times [\text{Development Process}] + K_{14} \times [\text{Project Domain}] + \ldots
\]
Solution

Define weight coefficients of subfactors
Solution

Define weight coefficients of subfactors

[Problem Solving] = \( N_1 \times \text{[Creativity]} + N_2 \times \text{[Zeal]} + \ldots \)

[Accountability] = \( M_1 \times \text{[End Result]} + M_2 \times \text{[People]} + \ldots \)
Solution

Define weight coefficients of subfactors

- Expert survey
- Calculate average/median
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

\[
(1 + K1 * \text{Accountability})
\]
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

\[
\text{Total Points} = \text{Know-How} \times \text{Problem Solving} \times (1 + K1 \times \text{Accountability}) \times (K2 \times \text{Performance} + K3 \times \text{Quality})
\]
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

Correct coefficients

Estimate factor ratings

Calculate Total Points

Compare results (apply common sense)

OK?

Yes

No

Correct coefficients
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

<table>
<thead>
<tr>
<th>Total points range</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 - 70</td>
<td>D1: Probationer</td>
</tr>
<tr>
<td>65 - 90</td>
<td>D2: Junior Developer</td>
</tr>
<tr>
<td>85 - 110</td>
<td>D3: Developer</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>200+</td>
<td>D7: Expert</td>
</tr>
</tbody>
</table>
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
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:
5. Discussion on developers efficiency evaluation: http://habrahabr.ru/post/101906/
7. 360 degree feedback: http://en.wikipedia.org/wiki/360-degree_feedback
Thanks!

Questions?

Valentin Anoprenko
anoprenko@devexperts.com