Output-based contracting in an Agile environment

by Nico Oosterom Product Owner at Glencore Agriculture & Richard Sweer Project Manager at Finidy

version 8 October 2019 – v1.1

Source: https://maritasoverpeinzingen.wordpress.com/tag/scrum/
Agenda

- Agile and commercial delivery models
- Core metrics – even for every Agile project/value streams
- The model – 4 KPIs with at least 35 underlying metrics

- Glencore Agriculture and our challenge
- Global contour of output-based contract
- How to monitor and control
- Our experiences with an output-based contract
Intro - Agile needs different commercial delivery models!

- Traditional hour tariff based
- Lower productivity and quality means higher revenue for supplier
- Many vendors ignore productivity and quality
- Most of the risk with customer, no control due to lack of metrics

- Outcome Based
  - Payment is tied to % of business benefits (reward sharing)
- Continuous Output-based
  - Supplier owns delivery and is paid per output delivered (e.g. Function Points)
  - Supplier is not paid if the quality requirements are not met (Definition of Done)
  - Higher productivity means more throughput and more profit (win-win scenario)
  - Most of risk with supplier
- Fixed Price
  - Fixed specification
  - Fixed price and mostly fixed date
  - Changes with a fee (mostly more than expected)
  - Risk with supplier and often also with customer (changes)
- Continuous Fact-based
  - Grip and control on costs, duration and quality by using metrics
  - Budget and duration risks with customer
  - Lower productivity and quality means higher revenue for supplier
- Time & Material
  - Traditional hour tariff based
  - Lower productivity and quality means higher revenue for supplier
  - Many vendors ignore productivity and quality
  - Most of the risk with customer, no control due to lack of metrics
Core metrics – Productivity, Cost effectiveness and Quality

- **Productivity rates** (Faster)
  
  \[
  \text{Hours of effort} \ \frac{\text{functional size/scope (e.g. Function Points) of delivered software}}{\text{functional size/scope (e.g. Function Points) of delivered software}}
  \]

- **Cost effectiveness** (Cheaper)
  
  \[
  \text{Project Euro cost} \ \frac{\text{functional size/scope (e.g. Function Points) of delivered software}}{\text{functional size/scope (e.g. Function Points) of delivered software}}
  \]

- **Product quality** (Better)
  
  \[
  \text{Defects} \ \frac{\text{functional size/scope (e.g. Function Points) of delivered software}}{\text{functional size/scope (e.g. Function Points) of delivered software}}
  \]
Core metrics – Size is a significant factor for effort and duration

1. **Project size/scope (Lines of code, Function Points, etc.)**
   - Effort for 1.000 FP project ~ 8 man-year\(^1\)
   - Effort for 10.000 FP project ~ 200 man-year\(^1\)

2. **Kind of software (factor 30-40)**
   - Nuclear power plant, air traffic system, bank system, etc.

3. **Effectiveness of individuals of team (factor 10-20)**

4. **Programming languages (factor 2-10)**
   - .NET, Java, Mendix, OutSystems, Thinkwise, Codeless, Angular, Polymer, Oracle, SQL Server, etc.

---

\(^1\) Prof. dr. C (Chris) Verhoef, Science of Computer Programming, volume 45, number 1, October 2002
Functional Size Measurement (FSM)
- ISO/IEC 14143-1:2006 Functional Size Measurements

Functional Size (FS)
- A size of the software derived from quantifying the Functional User Requirements

Nesma: High level counting is the ‘standard’

Examples
- ISO/IEC 24570:2018 NESMA (version 2.3)
- NESMA for software enhancement (version 2.3)
- ISO/IEC 29881:2010 FiSMA (version 1.1)
- ISO/IEC 20269:2009 IFPUG (version 4.3.1)
- ISO/IEC 19761:2017 COSMIC FFP (version 4.0.2)
- Interface points (Finidy)
Core metrics – Story- and Function Points can both be used

### Product Backlog
- Client prioritized product features

### Sprint Backlog
- Features assigned to Sprint
- Estimated by team
- Team commitment

### Backlog tasks

### Time-boxed "Sprint" Cycles

### Function Points
- Increase
- Decrease

### Sprint Planning Meeting
- Review Product Backlog
- Estimate Sprint Backlog
- Commit

### Daily Scrum Meetings
- Done since last meeting
- Plan for today
- Roadlocks?

### Sprint Review Meeting
- Demo features to all
- Retrospective on the Sprint

### Working Code Ready for Shipping (product)

### Scrum team
- Product Owner
- Scrum Master
- Development team

### Functional Size Measurement (FSM)
- Competitive productivity, cost effectiveness and product quality
The model – 4 KPIs with at least 35 underlying metrics

- **USP: increase success rate of a software IT project**
  - Combination of at least 35 metrics which makes it possible to increase the predictability of costs, turnaround time and quality of IT projects (Agile/Scrum, Kanban or Waterfall)
  - These metrics are divided into four areas
    - Better – Quality
    - Faster – Time to market
    - Happier – Satisfaction
    - Cheaper – Productivity

- **Supports two types of ‘appearances’**
  - Continuous Fact-based (calibrate the values of the KPIs/metrics)
    - 4 KPIs with at least 35 underlying metrics - paying Time and Materials
  - Continuous Output-based (basis for Output-based contracting)
    - 4 KPIs with at least 35 underlying metrics - paying price per **FSM (e.g. Function Points)**

---

*Cognizant is one of the world’s leading professional services companies, transforming clients’ business, operating and technology models for the digital era. Cognizant helps customer focus on their core and provides value by executing projects tied to output/outcome to clients’ business needs. As an example, Cognizant is helping a customer in The Netherlands to modernize their landscape by executing the program in Output-based Agile/Iterative model linked to Function Points delivered (based on NESMA standards)*

Saket Gulati – Head of Markets, The Netherlands (Cognizant).
The model - Metrics must be transparent and create a win-win

Calculation rules and guidelines of all used metrics/KPIs must be transparent (and published)
The model - Price per Function Point based on ± 35 metrics

Price/hours per FSM (e.g. Function Points)

1. Better (Quality)
2. Happier (Satisfaction)
3. Faster (Time to market)
4. Cheaper (Productivity)

Better
Happier
Faster
Cheaper
The model - Some examples of metrics

- Static source code quality (>300 sub-metrics by using tools)
- Software architecture quality/technical debt (by manual review)
- Functional code and decision coverage (measured by a tool) or per Function Point
- Defect removal efficiency metrics (measured per Function Point)
- Maximum amounts of ‘open’ defects per severity code in production per Function Point
- Etc.

1. Better (Quality)
2. Happier (Satisfaction)
3. Faster (Time to market)
4. Cheaper (Productivity)
The model - Some examples of metrics

- (Key)-user satisfaction (demo and after/every x period)
- Development team satisfaction
- Product Owner satisfaction (demo and after/every x period)
- Stakeholders satisfaction after/every x period
- Etc.

1. Better (Quality)
2. Happier (Satisfaction)
3. Faster (Time to market)
4. Cheaper (Productivity)
The model - Some examples of metrics

- Response and resolution time of defects per severity code in acceptance and production environment (MTTR and MTBF)
- Amounts of function points to deliver per sprint with one team
- Amounts of function points to deliver per sprint with \(<n\) teams
- Etc.
The model - Some examples of metrics

1. Better (Quality)
   - Price per function point with one team

2. Happier (Satisfaction)
   - Price per function point with <n> teams

3. Faster (Time to market)
   - Price per function points after implementing all the generators/generic components

4. Cheaper (Productivity)
   - Price per function point for module x
   - Etc.
The model – Start simple with a Excel sheet

### Better - code quality

<table>
<thead>
<tr>
<th>Metric</th>
<th>Target</th>
<th>Where</th>
<th>Average</th>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprint 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIOBE - Code Coverage</td>
<td>B</td>
<td>C</td>
<td>DoD</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>TIOBE - Abstract interpretations</td>
<td>C</td>
<td>B</td>
<td>DoD</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>TIOBE - Cyclomatic complexity</td>
<td>E</td>
<td>B</td>
<td>DoD</td>
<td>A</td>
<td>B</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>SonarQube - Bugs - reliability</td>
<td>B</td>
<td>A</td>
<td>DoD</td>
<td>C</td>
<td>C</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>SonarQube - Vulnerabilities - security</td>
<td>C</td>
<td>A</td>
<td>DoD</td>
<td>A</td>
<td>B</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

### Better - Defect metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Target</th>
<th>Where</th>
<th>Average</th>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprint 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum defect with severity 1 in first iteration of UAT (defects/FP)</td>
<td>1</td>
<td>UAT</td>
<td>1,667</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Maximum defects with severity 1 and 2 in Production (defects/FP)</td>
<td>2</td>
<td>Production</td>
<td>1,833</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Defect with severity 1 and 2 in code before production move</td>
<td>0</td>
<td>DoD</td>
<td>0,5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Happier - Satisfaction metrics (value between 1 and 10 where 10 is most satisfied)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Target</th>
<th>Where</th>
<th>Average</th>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprint 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development team happiness (per sprint)</td>
<td>7</td>
<td>Sprint review</td>
<td>6,75</td>
<td>7,5</td>
<td>7,0</td>
<td>6,0</td>
<td>7,0</td>
</tr>
<tr>
<td>Development team collaboration (per sprint)</td>
<td>7</td>
<td>Sprint review</td>
<td>6,75</td>
<td>8,5</td>
<td>8,0</td>
<td>6,0</td>
<td>5,0</td>
</tr>
<tr>
<td>Product Owner satisfaction (per sprint)</td>
<td>7</td>
<td>Sprint review</td>
<td>7,92</td>
<td>7,5</td>
<td>7,0</td>
<td>8,0</td>
<td>8,0</td>
</tr>
</tbody>
</table>

### Faster - through-put metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Target</th>
<th>Where</th>
<th>Average</th>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprint 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of FP per sprint</td>
<td>50</td>
<td>DoD</td>
<td>48</td>
<td>20</td>
<td>30</td>
<td>55</td>
<td>60</td>
</tr>
<tr>
<td>Minimum amount of Story Points/FP in stock</td>
<td>60</td>
<td>DoD</td>
<td>52,5</td>
<td>20</td>
<td>40</td>
<td>55</td>
<td>60</td>
</tr>
</tbody>
</table>

### Faster - response and resolution

<table>
<thead>
<tr>
<th>Metric</th>
<th>Target</th>
<th>Where</th>
<th>Average</th>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprint 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response &amp; resolution time for defects severity 1 (during UAT in days)</td>
<td>2</td>
<td>UAT</td>
<td>2,0</td>
<td>2,0</td>
<td>2,0</td>
<td>2,0</td>
<td>2,0</td>
</tr>
<tr>
<td>Response &amp; resolution time for defects severity 1 (during production in days)</td>
<td>1</td>
<td>Production</td>
<td>1,43</td>
<td>2,0</td>
<td>0,6</td>
<td>2,0</td>
<td>1,0</td>
</tr>
<tr>
<td>Response &amp; resolution time for defects severity 2 (during production in days)</td>
<td>2</td>
<td>Production</td>
<td>1,5</td>
<td>2,0</td>
<td>3,0</td>
<td>1,0</td>
<td>1,0</td>
</tr>
</tbody>
</table>

### Cheaper - productivity metrics (per sprint in hours/FP)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Target</th>
<th>Where</th>
<th>Average</th>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprint 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business analyst</td>
<td>4</td>
<td>DoD</td>
<td>4,7</td>
<td>6,0</td>
<td>5,0</td>
<td>5,0</td>
<td>3,5</td>
</tr>
<tr>
<td>Build (included unit tests)</td>
<td>10</td>
<td>DoD</td>
<td>8,3</td>
<td>7,0</td>
<td>8,0</td>
<td>10,0</td>
<td>8,0</td>
</tr>
<tr>
<td>Testing (view, API, E2E)</td>
<td>6</td>
<td>DoD</td>
<td>6,8</td>
<td>7,0</td>
<td>7,0</td>
<td>8,0</td>
<td>7,0</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td>DoD</td>
<td>19,8</td>
<td>20,0</td>
<td>20,0</td>
<td>23,0</td>
<td>18,5</td>
</tr>
</tbody>
</table>
Glencore Agriculture

Nico Oosterom
Agenda

• Introduction – what do we see in the market/problem statement
• Core metrics for every (agile) project and the role of Functional Size Measurement
• The model – 4 KPIs and 35-40 underlying metrics

• Glencore Agriculture and our challenge
• Global structure & content of our output-based contract
• How we monitor and control
• Our experiences with an output-based contract
Glencore Agriculture is a market leader in originating, handling, processing and marketing agricultural commodities, including grain, oilseeds, pulses, sugar, rice, cotton, vegetable oils, protein meals and biodiesel.
Our challenge is to modernize 8,000 Function Points

- Glencore Agriculture has written its existing Trading and Traffic System (ERP system) in Powerbuilder (Client) and PL-SQL/Oracle (Backend).
  - Powerbuilder is set to be replaced by a different programming environment
- The existing system is complex and comprehensive.
  - Used by over 900 users in more than 16 countries, covering the process from contract entry to invoicing
  - Scope of the system determined by conducting a NESMA 2.3 Function Point Analysis.
    Online scope (excluding batches and interfaces) is approximately 13,000 Function Points.
- After a proper preliminary study, we have chosen for the combination of Oracle (DBMS), Microsoft .NET and TypeScript/Angular as the new default platform.
- Rebuild while the shop is still open:
  - Domain driven / Modular rebuild
  - Keep using single Oracle database

The future of IT Cost Estimation
Trends for the new Decade
**Determine what’s in the price per FP – Deliverables & activities**

**Key success factors**

- Describe all the ‘system’ deliverables (the output) – based on PRINCE2 template (step 1)
- Determine what’s in the price per FP and what’s not in the price per FP (step 2 & step 3)
  - For each product and for each activity/Scrum ceremony
- Agree on the 4 main KPI’s (Better, faster etc.) and at least 35 underlying metrics

**BUILD-03: Source code**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Source code is a collection of files containing codes written in a specific language (e.g. Typescript, C#, PL/SQL) that is ‘translated’ into a set of instructions that can executed by an execution environment (e.g. JavaScript Engine, .NET, Oracle).</th>
</tr>
</thead>
</table>
| Composition | Source code is distinguished based on technology:  
Frond end (typescript / HTML 5 / SCSS)  
Back end (C#)  
Database (Oracle) |
| Appearance and format | IDE (Visual Studio, PL/SQL Developer). Source code is stored in TFS git repository. |

| Acceptance/quality criteria | Meet the Definition of Done  
Ticic TUCS ratings:  
- Abstract interpretation  
- Cyclomatic complexity  
- Compiler warnings  
- Code standards  
- Code duplication  
- Dead Code |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>‘system’ deliverables</th>
<th>Included in price/FP</th>
<th>Activities included in price/FP</th>
<th>Activities excluded from price/FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC-01: Project Start Architecture</td>
<td>YES</td>
<td>Generic components and accelerators</td>
<td>Coding and testing of PL/SQL programs</td>
</tr>
<tr>
<td>ARC-02: Project Architecture Variations</td>
<td>YES</td>
<td>Deployment (up to Acceptance)</td>
<td>Design, implementation or support of infrastructure</td>
</tr>
<tr>
<td>BA-01: Non-functional requirements</td>
<td>YES</td>
<td>Handover to line organization</td>
<td>Load, Stress and Performance tests</td>
</tr>
<tr>
<td>BA-02: User Stories</td>
<td>YES</td>
<td>Scrum Master and Project Management</td>
<td>Production Deployment</td>
</tr>
<tr>
<td>BA-03: UX design</td>
<td>NO</td>
<td>Support for solving defects</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Payments (price per FP) associated to clear output moments

Quality gate 1  
Definition of Ready

Sprint n-1  
Iteration n-1

Sprint n  
Iteration n

Sprint n+1  
Iteration n+1

Quality gate 2  
DoD + reviewing the code  
(architecture/technical debt)  
Pay 80% of the price per FP

Glencore Agriculture to review Sprint n or Iteration n

Definition of Done

Quality gate 3  
Release accepted  
Pay other 20% of the price per FP

Glencore Agriculture

4 Consecutive weeks in production with no defects

Glencore Agriculture

UAT/Exploratory testing by Glencore Agriculture

Live in production

Glencore Agriculture

The future of IT Cost Estimation  
Trends for the new Decade
Fact-based modernization with the use of Function Points

The future of IT Cost Estimation
Trends for the new Decade
Fact-based progress with the use of Function Points

The future of IT Cost Estimation
Trends for the new Decade
Work excluded from price per FP is calculated as ‘virtual’ FP’s

- We have a bonus/Malus on throughput (# Function Points per period)
- Throughput is based on ‘real’ Function Points and ‘virtual’ Function Points
  - Not everything can be counted as Function Points – framework upgrades, downtime etc.
  - ‘Virtual’ Function Points are RFC hours divided by agreed productivity
  - In example below (using 20 hours per FP) the throughput is 250+10=260 Function Points

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Start date</th>
<th>End date</th>
<th>‘real’ Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 6</td>
<td>Mon 3 Dec 2018</td>
<td>Fri 11 Jan 2019</td>
<td>105 FP</td>
</tr>
<tr>
<td>Iteration 7</td>
<td>Mon 14 Jan 2019</td>
<td>Fri 1 Mar 2019</td>
<td>145 FP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>250 FP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RFC</th>
<th>Description</th>
<th>Hours</th>
<th>‘Virtual’ Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC-2019-01</td>
<td>Iteration 6 - downtime</td>
<td>80</td>
<td>4 FP</td>
</tr>
<tr>
<td>RFC-2019-02</td>
<td>Upgrade AG-grid</td>
<td>120</td>
<td>6 FP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10 FP</td>
</tr>
</tbody>
</table>
Measure code quality by SonarQube/Tiobe is not enough ...

- You will need manual review too
Make sure Technical Debt is captured, visible and planned!
QA metrics have to be fully integrated in the build pipelines.
Aim for ‘zero tolerance’: monitor # of defects in UAT&PRD
Our experience with an output-based contract

- Explicit demand for price per Function Point brought different players to the table, some vendors stepped back
- Importance of very detailed product descriptions (system deliverables)
  Crucial to have very clear what’s in the price per FP, saves a lot of discussion!
- Vendor should make thorough assessment to understand potential complexity.
- Don’t bother dev team(s) with the commercial side, quality first
- Measured values of the KPIs/metrics reflects the reality
  - Based on the quality metrics, we notice when new employees are added to the project
  - Process ‘bottlenecks’ become visible (for example # FP’s in stock)
  - Enable tooling to closely monitor where possible
- Helps to get focus for any improvement in all area’s (automation etc.)
Questions?

Nico Oosterom
Nico.Oosterom@glencore.com
www.glencoreagriculture.com

Richard Sweer
Richard@finidy.nl
www.finidy.nl

Download here the whitepaper
More grip less guesswork in agile IT projects

The future of IT Cost Estimation
Trends for the new Decade