Claas Augner

A practical model of Software Telemetry

Addressing the need for knowledge about remotely deployed systems

Master thesis
Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion
Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion

Motivation
Research questions
Procedure
1. Introduction

Motivation:
- Personal: frustration & scepticism as user + developer view
- Software ≈ result of customer-supplier agreement → requirements
- Problem 1: SW requirements/implementations often imperfect
- Problem 2: After deployment → only explicit customer feedback

Solution:
Software Telemetry (my definition) = “systematic collection, transmission and processing of selective information from a deployed software system, with the objective of acquiring actionable insights about the operation of the system”
1. Introduction

Premise:
Knowledge about deployed software systems is important.

Research Problem:
Knowledge is difficult to obtain, if systems are remotely deployed.

1\textsuperscript{st} Research Question: What are practices to obtain this knowledge?
2\textsuperscript{nd} Research Question: How can these practices be modelled?

Procedure:
- Literature review → Background
- Review of software products → Practices
- Scientific modelling → Model
- Case study → Model instance
- Qualitative evaluation → Model evaluation
Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion
2. Background

Historical context:
- “telemetry” ≈ “measuring at a distance”
- “Software Telemetry” – adopted by Microsoft/Mozilla
- Roots: Software Measurement, SW Monitoring (since 1960s)

Contemporary context:
Analytics Instrumentation Performance Evaluation
Error Reporting Logging Usage Data Collection

Theoretical underpinnings:
No SW Telemetry theory yet. Related theory:
- Software Maintenance (context)
- Software Operation Knowledge (artefact)
- Software Quality (goal)
2. Background

Research gap:
Similar research has been conducted. But: No general research on practical applications yet. And: No focus on remotely deployed applications yet.

Significance:
- Increasing complexity
- Increasing user expectations
- Paradigm shift from agile to Continuous Deployment
- Remote deployments
- SW maintenance longest phase
Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion

Procedure
Choice
Examples
3. Practices

Procedure:
Inclusion criteria → review dimensions
→ software selection → sources identification
→ review → integration of results

Inclusion criteria:
– SW must apply telemetry.
– SW must be open-source or telemetry well-documented.
– Selection should be diverse.

Review dimensions:
About    Goals    Collection    User involvement
Sources  Scopes    Transmission  Privacy / Security
Overview Architecture Processing
3. Practices

11 Software products:

- ATOM
- Eclipse
- Chrome
- Firefox
- Nextcloud
- Subversion
- FAIRPHONE
- Ubuntu
- Windows

Primary sources: software, source code, documentation, etc.
Secondary sources: literature, bug tracker, etc.
3. Practices

Example 1: 🌮 ATOM

---

Claas Augner: *A practical model of Software Telemetry.*

Freie Universität Berlin

04.05.2017
3. Practices

Example 2: [Nextcloud]

Usage report

You can help us to improve Nextcloud by sending us some data about your current setup and usage.

We take your privacy seriously. The data are anonymized and you can enable/disable it at any time, by default it is always disabled. Below you can also adjust what kind of data are send and always see the last report send to us.

When the server receives a new report of your instance, all entries from previous reports are removed. In case you disable one of the settings below, you can send a new report, to delete the data, that is currently stored on the server.

- Send new report now
- Send "Usage report" monthly

Data to send

- Server instance details (version, memcache used, locking/previews/avatars enabled?)
- PHP environment (version, memory limit, max. execution time, max. file size)
- Database environment (type, version, database size)
- App list (for each app: name, version, is enabled?)
- Static (number of files, users, storages per type)
- Number of shares (per type and permission setting)
- Encryption Information (is it enabled?, what is the default module)

Last report

```
{
    "id": "526a8d1552887",
    "items": []
    [
        "server",
        "version",
        "9.1.15"
    ]
```

Sent on: 4. Oktober 2016
3. Practices

Example 3: Firefox

![Firefox Telemetry Interface](image)

Telemetry Data
This page shows the information about performance, hardware, usage and customizations collected by Telemetry. This information is submitted to Mozilla to help improve Mozilla Firefox.

- General Data
- Environment Data
- Session Information
- Scalars
- Keyed Scalars
- Histograms
- Keyed Histograms
- Simple Measurements
- Telemetry Log
- Slow SQL Statements
- Browser Hangs
- Thread Hangs
- Late Writes
- Add-on Details
- Histograms Collected by Add-ons

The Firefox Health Report is a tool that helps users understand their browser performance and share data with Mozilla. It includes options for enabling health reports, sharing additional data, and allowing Firefox to send backlogged crash reports on the user's behalf.

Example usage:

```
about:telemetry
about:healthreport
```

Firefox automatically sends some data to Mozilla so that we can improve your experience.

Choose What I Share

Firefox Health Report

- Version: 54.0.1
- Update channel: nightly
- Updates: automatic

- This Month
  - Total sessions: 33
  - Time open: 2 days
  - Application crashes: 2
  - Plug-in crashes: 0
  - Active Add-ons
    - Extensions: 20
    - Plugins: 3

Startup Time by Day

- Time to start: 20 seconds
- Time to open: 2 days

Tips
- It looks like you've used Firefox recently. To learn more:
  - Visit our support website.
Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion

Telemetry system
Telemetry scopes
Telemetry model
4. Model

<table>
<thead>
<tr>
<th>Artefacts</th>
<th>Scopes</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data</td>
<td>Information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Knowledge</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Collection</th>
<th>Transmission</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategies</th>
<th>Scopes</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User involvement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Privacy / Security</td>
<td></td>
</tr>
</tbody>
</table>
4. Model

Telemetry scopes

<table>
<thead>
<tr>
<th>External Configuration</th>
<th>Internal State</th>
<th>External Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Input)</td>
<td>(Behaviour)</td>
<td>(Output)</td>
</tr>
</tbody>
</table>

- Internal Usage
- Internal Performance
- Internal Validation
- External Usage
- External Performance
- External Validation
4. Model

Telemetry model

Collection
- Strategy
  - Sender-Push
  - Receiver-Pull
- Granularity
  - Raw collection
  - Aggregate collection
- Data management
  - Memory
  - Database
  - Files

Transmission
- Mode
  - Automatic
  - Semi-automatic
  - Manual
  - Unidirectional
  - Bidirectional
- Strategy
- Direction
  - Text
  - Binary
  - HTTP
  - Other
- Data management
  - Database
  - Files

Processing
- Artefacts
  - Aggregates
  - Alerts
  - Reports
  - Visualisations
4. Model

Telemetry model

User involvement

Consent
- Consent prompt
- Transmission preview
- Transmission log
- Settings view
- Active consent
- Passive consent
- Interaction start
- Before collection
- Before transmission
- Permanent
- Temporary
- One-time
- Boolean choice
- Ordinal choice
- Nominal choice

Privacy
- Data avoidance
- Data minimisation
- Data retention
- User control
- User transparency

Security
- Confidentiality
- Integrity
- Non-repudiation
- Accountability
- Authenticity
Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion

Context / Objectives
Architecture
Implementation
5. Case Study

Context:
- *Acme GmbH* = German medium-sized company with 1 SW product
- *ABC* = web application for enterprises, modularly structured, highly configurable and largely extensible

Objectives:
- General: Verify if model applicable to a real-world SW product.
- Author: Flexibility, user transparency, developer usability.

Architecture:
- 2 services (collection + transmission) + a Telemetry façade
- independent processing server
- Approach: New probes must be described in a Java *enum*
5. Case Study

Telemetry model

Collection
- Strategy
  - Sender-Push
  - Receiver-Pull
- Granularity
  - Raw collection
  - Aggregate collection
  - Memory
- Data management
  - Database
  - Files

Transmission
- Strategy
  - Automatic
  - Semi-automatic
  - Manual
- Mode
- Direction
  - Unidirectional
  - Bidirectional
- Format
  - Text
  - Binary
  - HTTP
  - Other
- Channel
  - Database
  - Files

Processing
- Data management
- Artefacts
  - Aggregates
  - Alerts
  - Reports
  - Visualisations
5. Case Study
Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion
6. Evaluation

Data assessment:
- **Goal**: Functional / data evaluation
- **Procedure**: Insert 10 probes with hypothesis → run tests → verify

Experiment:
- **Goal**: Practical / developer evaluation
- **Procedure**: Test and survey with 3 developers

Results:
- Confirmed flexibility of the model.
- Confirmed functionality of all components.
- Provided evidence for practicability, effectiveness and usability.
Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation

7. Conclusion

Limitations
Results
Future research
7. Conclusion

Limitations:
- Bias towards open source software.
- No Software Telemetry processes.
- No actual user involvement.
- No quantitative evaluation.

Results:
- Practices are diverse, but follow recurring patterns.
- Obtaining knowledge about remotely deployed systems is doable.
- Model provides basis for successful implementation.

Future research:
- Refine with regard to limitations.
- Study discontinuation examples of SW Telemetry.
- Study user perception of SW Telemetry.
Thank you for your attention.

A practical model of Software Telemetry

Addressing the need for knowledge about remotely deployed systems

Any questions?