SOFTWARE INTEROPERABILITY ANALYSIS IN PRACTICE

This briefing reports evidence on the current difficulties faced by practitioners in performing effective and efficient software interoperability analysis based on scientific evidence from a survey study.

FINDINGS

The findings presented in this briefing define software interoperability analysis as the process of checking the constraints and assumptions of two software units in order to find if they have any mismatches that impede their desired communication.

The as-is state of practice for software interoperability analysis shows that a large group of practitioners (40.2%) neglect to perform interoperability analysis mainly due to the lack of knowledge and experience about it and due to task prioritization reasons.

Analysis results, for data collected from those who actually performed interoperability analysis, indicate that the current performed interoperability analysis is immature, unstandardized, and unsystematic.

The findings describe that performing interoperability analysis is of high cost (mostly reported to range from 10% to 30% of the project cost). However, not performing it and facing unexpected mismatched later in the project is even more expensive (mostly reported to range from 10% to 50% of the project cost).

The evidence shows a significantly perceived need for better practical support to overcome practical difficulties that include the following:

- Lack of focus on detecting “conceptual” mismatches compared to “technical” mismatches (39.1%)
- Lack of support for traceability between interoperability analysis activities and results within a project and among projects (37.5%)
- Lack of interoperability analysis guidelines and best practices for practitioners (40.6%)
- Manual effort in analyzing the external software units and in documenting the analysis results (40.6%)

The evidence reveals a perceived need for enhancing the content of the input artifact, which is used in software interoperability analysis, including:

- Communication constraints like network protocols, message formats, etc. (57.8%)
- Semantic constraints like glossaries, goals, rationale, etc. (48.4%)
- High-level architecture view like architecture style and design patterns, etc. (60.9%)
- Behavior constraints like pre/post conditions, interaction protocols, control flow, etc. (66.2%)

The survey results also present a significant perceived need to enhance the presentation of information in the input artifact used in interoperability analysis. This includes overcoming the following problems:

- Mixing conceptual and technical constraints without clear borders between them (46.9%)
- Unstructured verbose text (39.1%)
- Lack of easy-to-read process diagrams like flow charts (46.9%)
- Inconsistency in reporting constraints for the different data items and services (42.2%)

Who is this briefing for?

Software engineering practitioners who want to know about the problems experienced when analyzing the interoperability between software units within integration projects based on scientific evidence.

Where the findings come from?

All findings of this briefing were extracted from the online survey conducted by Hadil Abukwaik and Dieter Rombach.

What is included in this briefing?

The main findings of the conducted survey with software engineers with software integration experiences in the industry about the faced difficulties in performing software interoperability analysis.

What is not included in this briefing?

Additional information not presented in the original survey paper.

Detailed descriptions about the original survey design and execution.

To access other evidence briefings on software engineering:

http://ease2017.bth.se/

For additional information about AGSE (Software Engineering Research Group: Process and Measurement:

http://agse.cs.uni-kl.de/

ORIGINAL RESEARCH REFERENCE

Hadil Abukwaik and Dieter Rombach. ***Software Interoperability Analysis in Practice – a Survey***. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17). ACM, New York, NY, USA, 12-20. DOI: https://doi.org/10.1145/3084226.3084255