Project no.:	H2020-MSCA-RISE-2017-778233
Project full title:	Behavioural Application Program Interfaces
Project Acronym:	BEHAPI
Deliverable no.:	D.2.1 (M6)
Title of Deliverable:	Portfolio of use-cases and of current practices for API provisioning.
Work package:	WP2 (task 2.1)
Type:	R
Lead Beneficiary:	**UNIVERSITÀ DEGLI STUDI DI TORINO (BEN9, UNITO)**
Dissemination Level:	PU
Number of pages:	
Contract Delivery due date: 31/08/2018 (M6)	
Actual delivery date of version n.1: 31/08/2018 (M6)	
Actual delivery date of version n.2: 22/10/2018 (M8)	

Acknowledgement: This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 778233

Abstract:

This deliverable reports on the results of the task "T.2.1: Current practices and state-of-the-art of Application Program Interface (API) provision" of the work package (WP) 2 "API Provision". The goals of WP 2 are to develop and enhance languages tailored to the rigorous and human friendly description of behavioral APIS (b-APIs). Activities in WP 2 are tailored to the definition of techniques to (semi-)automatically generate b-API descriptions from source code; to develop validation and verification techniques to assure that the documentation of a b-API is aligned with its implementation, in both its functional and non-functional requirements. The task T.2.1. was designed in order to address the Objective O.2.1 "Surveying current practice in API provision", i.e., to analyse the problems that providers face when creating and managing APIs; spelling out limitations of current approaches and technologies for API development; and identifying opportunities for improvements in the software development process.

The information reported in this document has been collected during August 2018 (M6) and based on the answers provided by the following 7 (seven) non-academic partners of the project, which are API providers.

- ACTYX AG (BEN 10, ACT),
- BITLAND SRL (BEN 11, BTL),
- IXARIS SYSTEMS (MALTA) LIMITED (BEN 12, IXR)
- GREEN BY WEB LDA (BEN 13, GBW)
- XIBIS LIMITED (BEN 15, XIB),
- DCR SOLUTIONS (BEN 16, DCR),
- MCAFEE ARGENTINA S.A (BEN 22, MCF)

This document summarises the form used for gathering information and the collected data. The structure of the survey has been conceived and implemented as a Google form through a series of teleconferences by the following academic partners of the project:

- UNIVERSITÀ DEGLI STUDI DI TORINO (BEN 9, UNITO),
- UNIVERSITY OF LEICESTER (BEN 2, ULEIC)
- UNIVERSITY OF KENT (BEN 4, UKENT),
- UNIVERSIDAD DE BUENOS AIRES (BEN 20, UBA)

Part of the activities reported in this document have been carried out during the following secondments:

Fellow Id	Category	Declaration Number	Person/Month
3	ER	4	0.5
4	ER	5	0.3
5	ESR	6	0.4
6	ER	7	0.3

Table of content

1. Introduction

This deliverable reports on the activities carried out in order to collect information from the partners of our consortium about the current practices on the provision of APIs. The goal of this activity is to identify which partners develop and provide APIs for either internal or external consumption to understand the different application domains in which APIs are used, and the general aspects of the developed APIs, such as their structure, complexity and availability (i.e., whether they are publicly available or not).

Additionally, we are aimed at surveying the state of the art of the current development processes followed by our industrial partners when delivering APIs. For this reason, we gather information about the current practices on the specification and documentation of APIs, the technologies and languages used for implementing them, and the approaches applied for validation and verification.

Finally, we also address the question of assessing those aspects that are identified as advantages of the specification and development APIs (related, for instance, to costs and quality attributes such as verifiability, robustness, reliability, etc.), their potential disadvantages (e.g., costs, in the development, maintenance, increased constraints, negligible return, etc.). We also collect information about interests (technical and economic) of partners about possible future developments of new services and/or extending those are already provided by relying on programmable APIs. We collect information about the motivations, goals and priorities that such activities may have. Also, we gather information about the interests and possibility for developing, applying or extending approaches to validation and verification, such as testing, run-time monitoring, formal verification, of programmable APIs. We additionally, focus on the necessity of properly addressing non-functional requirements, such as quantitative properties of the services like bandwidth, security, number of requests per time unit.

The information was collected through the online form described in Section 2. The collected results are presented in Section 3. The analysis of the information and conclusions are drawn in Section 4.

2. Survey structure and methodology

Considering the geographical distribution of the non-academic partners, the different degrees of involvement of the partners in the BehAPI consortium and the particular time of the year during which the information was supposed to be collected (the first 3 weeks of August 2018, M6), it was decided to propose the survey as an online form that partners could easily and independently reach and complete with as little overhead as possible from their side and as little assistance as possible from the academic partners. We opted for using Google Forms, a customizable polling service that Google freely offers, because of its reliability, ease of use and availability of facilities for gathering and organizing the collected data into spreadsheets. In order to reduce the effort required by the non-academic partners for filling out the form and to simplify the reporting of the collected information, it was agreed that the form would have been structured as a series of detailed questions on the current practices of API provisioning and, wherever possible, a limited set of predefined answers would have been made available.

Of all the non-academic partners involved in the project, the following ones were asked to participate in the survey: ACT, BTL, IXR, GBW, XIB, DCR, MCF.

Below we reproduce the form used for eliciting information from partners. The form has been designed in order to get information about the nature of the APIs developed by the partners and whether they are publicly or not, current practices, techniques and technology used, limitations and strengths of current practices.

Template used for the survey

BehAPI survey on practices for API provision

This document's aim is to elicit information on current practices concerning the production and consumption of APIs from academic/industrial partners of BehAPI.

Background information and domain

1 - *Name and application domain (eg., public, research, development) of industrial partner*

Current practices on API provision

By API provision we mean developing services with programmable API so that functionalities provided by these services can be (re)used by applications running within or outside the organization.

2 - Are you producing/exposing APIs at all, are these for internal or external use (or both)?

- If yes, please give a few examples and continue to the next question.
- If no, please jump to question 8

3 - We are looking at the API you produce and try to understand their **complexity**.

- Do they support different kinds of operations? Eg., log-in, request data, etc.
- Is there any expected order of execution of these operations? Eg., log-in should happen before data requests.

If you answered yes to any of the above, please give some examples.

4 - Concerning the technical aspects of the API you produce:

- How are API specified and documented?
- Which languages and technologies are used to develop and run/expose them?

By specification we intend any formal or informal description of the API, which operations it provides, how it can be used, which features it exposes, etc.

5 - Are you using any Verification and Validation (V&V) approach like testing, runtime monitoring, formal verification to verify or validate of your APIs? If yes, please give a brief description of the practices you use.

6 - Do you have a catalogue of the APIs you provide? • If so, is it public? • If not, would it be possible to make it available to some or all partners of the projects? Under which conditions (eg., non-disclosure agreement) ?

7 - What are the advantages of the practices you are currently adopting in the specification and development of API? What are their limitations or the challenges/problems you perceive? Some examples of advantages and disadvantages: development costs, easy verification, easier or more difficult maintenance, etc.

Potential for future development and provision via API.

8 - Is there interest and possibility (technical and economic) to develop services with programmable API or extend those you already provide? • If yes, provide information on the motivations, goals and priorities for this effort.

9 - Is there interest and possibility (technical and economic) to develop or extend approaches to documentation, V&V (testing, run-time monitoring, formal verification) of programmable APIs? • If yes, please, comment on what are the interests and future goals, and their priority.

10- If you answered yes to either 8 or 9: are there specific desiderata as to the quantitative properties of the services you expose or wish to expose via APIs (eg., bandwidth, security, number of requests per time unit). Could you make some examples for each desideratum?

3. Summary of collected data

After the agreed deadline for the compilation of the survey had passed, qualitative information gathered from the partners who participated in the survey was accessed via Google services in the form of a Google sheet and manually analysed by the academic partners. Below is a summary of the collected data, possibly adjusted to make it suitable for presentation in this deliverable.

3.1 Application Domain of the provided APIs

3.1.1 Domains

The information in this deliverable corresponds to organisations whose main activities are in the following application domains:

- Industrial Software
- Fintech
- Cybersecurity
- Research and Development
- Public sector
- IT company

3.1.2 Provision for internal/external usage

All of the non-academic partners involved in the survey stated that they rely on APIs provision for internal usage. Moreover, **nearly 70%** of the industrial partners provide APIs for external use, i.e., to be consumed by organisations that lie outside their own organisation boundaries.

3.1.3 Use cases of API provisioning

In this section we give an overview of the APIs developed by the surveyed partners. The level of detail differs depending on whether their API's and/or their documentation are publicly available or not. Below are the collected examples of APIs, none of which is publicly available:

- APIs for accessing domain objects (such as machine parameters, manufacturing orders);

- APIs corresponding to typed event streams for collecting notifications on domain object state changes;
- APIs for creating and issuing virtual credit cards;
- APIs for querying account balance;
- APIs for user/client authentication.

In one case, a non-academic partner has provided extensive information on the use of (publicly available) APIs for accessing and using Dynamic Condition Response graphs (DCR for short), a constraint-based graphical notation for processes addressing Adaptive Case Management (i.e., the coordination of work that is not routine and predictable and usually requires human coordination[1]). Below we report the collected information.

The provided APIs are described in the public catalogue *DCR Active Repository*[2] enabling third party applications to use DCR Graphs as repository for storing and executing graphs. DCR Active Repository is an API that provides access to DCR engine from external Enterprise Content Management (ECM). DCR Active Repository enables third party applications to use DCR Graphs for handling business processes by storing and executing graphs.

DCR Graphs are represented using DCR XML[3] which is the output of the DCRGraphs.net designer tool. DCR XML is used for representing classes and instances of processes. DCR graph contains the information about the events such as event descriptions, roles assigned to events, and their states(e.g: pending, enabled, executed).

This API is publicly available and its description can be found at http://wiki.dcrgraphs.net/. Here we just reproduced some elements.

1) DCR graphs
   http://www.dcrgraphs.net/api/graphs
   GET
   *List URIs of graphs, id and titles in collection. Filter options exists to search/filter specific graphs.
   Optional parameters are:

[1] Hamid R. Motahari Nezhad, Keith D. Swenson: Adaptive Case Management: Overview and Research Challenges. CBI 2013: 264-269. 2012
[2] http://wiki.dcrgraphs.net/wiki/50/dcr-active-repository.
[3] DCR XML is described in the article Exformatics declarative case management workflows as dcr graphs by Tijs Slaats, Raghava Rao Mukkamala, Thomas Hildebrandt, and Morten Marquard. In Business Process Management, volume 8094 of Lecture Notes in Computer Science, pages 339–354. Springer Berlin Heidelberg, 2013.

title=title - will search for graphs containing "title" in title
sort=title (default) - sort graphs by title or other fields
orderby=asc(default)
sort: can be title, id, created, modified
orderby: asc or desc

PUT

N/A

POST

*Create a new DCR graph, returning the id. A DCR Graph is described using DCR XML

DELETE

N/A

2) DCR graph
http://www.dcrgraphs.net/api/graphs/id

GET

*Retrieve the DCR graph represented as DCR XML

PUT

*Update the DCR graph using DCR XML

POST

N/A

DELETE

Delete the graph

3) DCR graph instances
http://www.dcrgraphs.net/api/graphs/id/sims

GET

*Retrieve the DCR graph instances, i.e. instance id (simid) and title. Various filters exists to retrieve representative and swimlane graphs:
isScenario=true or false - if true returns only scenarios (representative graphs)
Format=DCRXMLLog - optional,
only used if DCR XML Log format is needed
Filename=log.xml - optional filename

PUT

N/A

POST

*Instantiate a DCR graph returning a new instance ID (simid). An optional Title of the simulation can be provided.

API enhanced and now support creating multiple instances by important DCR XML Log which creates multiple instances at the same time. Use parameter Log to enable import.

DELETE

Delete the graph

4) DCR graph instance events
http://www.dcrgraphs.net/api/graphs/id/sims/simid/events

GET

*Retrieve the events of the DCR graph instance
Filter must be provided to search/filter specific instances
filter=all
filter=enabled-or-pending
filter=only-enabled
filter=only-pending

PUT

N/A

POST

Create a new event in the graph

DELETE

N/A

3.1.4 API Structure and Complexity

All of the provided examples of APIs support different operations, like log-in, data request, etc.. Moreover, **more than 60%** of them constraint the order in which operations should be performed. Below are some examples.

- Events are expected to obey causal relationships, for instance:
  - states cannot be updated before they are created;
  - domain objects validate —and possibly reject— incoming commands based on their state, i.e., some operations can be executed only when the domain object is in some state that allows the execution of the operation.
- Operations can be executed only by authorised clients. In such cases, the authentication methods need to be executed before the requested operations on the API.

  - In what concerns authentication/authorisation, JWT tokens[4] and Role-Based Access Control (RBAC) are used to validate which data can be sent to which client.
- Initializers are required to be executed before methods that modify the state of a graph (e.g.: ExcuteEvent, AdvanceTime).
- Domain constraints on IoT devices should be observed when performing operations. For instance, in an IoT application in an irrigation scenario, a garden should be created first; then the corresponding IoT device need to be registered and activated to for a new user; then associated with the newly created garden. Only after that, the irrigation schedule can be programmed.

3.2 Current practices

3.2.1 Approaches to specification

The surveyed non-academic partners of the consortium have reported the following methods and techniques for specifying and documenting APIs (the quantity of partners providing a particular answer is indicated between parenthesis, note that partners provided multiple answers).

- Informal textual description, possibly as a markdown document (3);
- Self developed documentation language (http://wiki.dcrgraphs.net/wiki/50/dcr-active-repository) (1);
- Informal comments in the source code (1);
- Data Types (3):
  - For input and output parameters of operations (using an Hypertext Transfer Protocol (HTTP) request Integrated Development Environment) (1);
  - Static (i.e. non-behavioral) data types for possible events in streams (1);
  - no behavioural data type for the specification of constraints among operations (1);
- RESTful API Modeling Language (RAML) and Swagger/Open API (for REST APIs) (3);
- HeadRest, a formal language for API description[5] (1).

[4] JWT stands for "JSON Web Tokens" and JSON is the acronym for "JavaScript Object Notation", a common notation for specifying records of data in JavaScript code running within Web pages.
[5] Vasco T. Vasconcelos, Antonia Lopes, and Francisco Martins. HeadREST: A Specification Language for RESTful APIs. http://rss.di.fc.ul.pt/wp-content/uploads/2018/05/main.pdf.

3.2.2 Languages and technologies

The surveyed partners of the consortium have reported the following technologies and languages for implementing APIs (the quantity of partners providing a particular answer is indicated between parenthesis, note that partners provided multiple answers).

- Java (4);
- Scala (2);
- Hypertext Preprocessor PHP (1);
- NODEJS and associated frameworks (1);
- JavaScript/Typescript (1);
- Python (1);
- Microsoft .Net Web API + Entity Framework (1);
- ASP.NET on IIS (1).

3.2.3 Current practices on Validation and Verification

The surveyed non-academic partners of the consortium have reported the following techniques and methodologies to be currently in use for the validation and verification of APIs (the quantity of partners providing a particular answer is indicated between parenthesis, note that partners provided multiple answers).

- Testing (7)
  - Backend test for the verification of API's result (1);
  - Black-box unit testing based on fixed scenarios, hand-coding the behavioral semantics (1);
  - Automated Testing, Based on Behave framework (1): https://github.com/behave/behave.
  - Internally developed framework (1);
  - Automated Postman testing, to run test scripts and verify the test cases specified on each API (1);
  - Testing using RestEasy (1);
  - Integration tests on API operations (1).
- Monitoring of API's backend through specific monitors based on custom code (1).

3.3 Assessment of API specification and development

3.3.1 Advantages of API specification and development

The surveyed partners in the consortium have identified the following as main advantages on the specification of APIs and their development (the quantity of partners providing a particular answer is indicated between parenthesis, note that partners provided multiple answers):

- Solid specification approaches are seen as a key factor that could **improve the quality of delivered software** (7), in the following dimensions
    - Understandability (2)
    - Documentation (3)

- **Easy of integration** (6) with other clients, backend and components and reliability of implementation. For example, when new APIs are developed in a project, usually differences from expectations are encountered when delivering/integrating the API. Having solid method of specifications and the ability to automatically test against that specification, would help in reducing those errors.

- **Verifiability** (4): APIs improves the way in which coding and testing is structured
    - Automated behavior testing of command APIs (3)

- **Maintenance** (2):
    - Long-term compatibility —not only structural but also behavioural— before deploying a change;

- **Code generation** (1): APIs specifications may allow for the automatic generation of code of applications.

3.3.2 Disadvantages of API specification and development

Despite the advantages associated to having accurate and precise description of APIs, the surveyed partners in the consortium have identified the following issues (the quantity of partners providing a particular answer is indicated between parenthesis, note that partners provided multiple answers):

- Specification is difficult and time consuming (3):
  - Needed developers with advanced skills and academic formation (2).
  - Unfamiliarity (1). Getting partners to work with new unfamiliar technologies can be difficult.
- Maintenance (4):
  - Effort to keep code and spec in sync (3); since it is hard to keep the specification updated with the changes suffered by the API, especially when the APIs are being developed.
  - Constant upgrades and changes affect the repository (1), but they do not pertain the APIs. It is increasingly hard to maintain such a tightly integrated code.
- Effort to make specification machine-readable (1), but it is expected that to be outweighed by forcing the specification to remain up-to-date.

3.4 Future plans and expectations

3.4.1 Foreseen extensions of provided APIs

All but one surveyed partners foresee changes, adaptation and extensions on the APIs they provide. The main motivations are (the quantity of partners providing a particular answer is indicated between parenthesis):

- Business reasons (3):
  - the domain model is constantly growing due to economic needs, which requires to provide attractive solutions to real customer problems (2).
  - new clients demand new requirements and features that need to be implemented (1).
- Nature of the developed software: solutions are built completely on APIs and they are continuously updated and improved (1).
- Integration with 3rd party services to allow customer to customize their applications (1).

3.4.2 Interest on new development approaches

The surveyed partners in the consortium highlight the following main expectations regarding to the development process of APIs (the quantity of

partners providing a particular answer is indicated between parenthesis, note that partners provided multiple answers):

- Testing techniques (4):
  - Enhancing automated test suite generation to achieve better testing quality (3).
  - Address contract-testing frameworks (1).
  - Tests for documentation (1).
- API verification (formal approaches for specific applications) (3).
- Run-time monitoring in specific projects (1).
  - Monitoring of traffic for availability of service (including customer-side issues) and service reporting.
- QoS assessment, for instance, ensuring number of requests per time unit in order to handle, e.g., peeks of usage (5).
- Principled approach (e.g., behavioural specification) for managing the complexity of a growing domain model (1).

3.4.3 Need of handling quantitative properties of services

The surveyed partners have identified aspects of APIs that require proper approaches to deal with quantitative properties of systems. Such needs impact on several activities of the development process, from specification to implementation and verification and validation. The examples for the considered application domains are the following:

- throttling requirements (e.g., number of requests per time unit) (2).
- Role-Based Access Control (RBAC) and security requirements (2).
- Monitoring of traffic to deal with availability of service (including customer-side issues) and service reporting (1).
- Device capabilities, i.e., services are provided and consumed over a peer-to-peer mesh network, where each node is a rather low-powered edge device (a tablet or IoT gateway). The frequency of computation events ranges from infrequent human input (every few hours) to machine sensors (multiple values every second) (1).
- Sensitivity of information handled by the API (1).
- Reasoning on bandwidth consumption (1).
- Reasoning about requests per hour and response time and standard deviation for response times (3).
- Techniques for populating system with realistic data, e.g. 50,000 users in a database in order to reason about quantitative properties (1).

4. Conclusions

This deliverable reports on the activity performed in the project in order to collect information about the APIs used and possibly developed by the industrial partners in the consortium. The surveyed partners cover 6 different application domains and rely on APIs not only for internal usage but also provide them to 3rd party organisations.

Based on the outcome of the survey, we can draw the following considerations:

- APIs form an essential part of every complex software infrastructure. This fact substantiates the potential impact of the BehAPI research project in the future practices for software development and maintenance.
- Despite the prominent role of APIs, in most cases they are only informally documented. This practice hinders the use of automated tools supporting software architects and developers, which would be extremely valuable. Indeed, all partners have provided examples showing that the operations on their APIs require them to be executing observing some order or constraint about causal dependencies.
- The impact of APIs on software correctness and the importance of API validation and verification are witnessed by the extensive use of various testing and runtime monitoring techniques adopted by the surveyed partners. In general, there is strong agreement about the importance of relying on standard specification of APIs and their benefits.
- There are very few documented cases of static analysis concerning APIs. This is justified in part by the lack of widespread techniques and adequate supporting tools, in part by the need of personnel (designers and developers) with adequate professional skills. In this sense, the aims of the BehAPI project strengthen existing dynamic techniques, as well as helping filling the gap concerning the static ones.
- There is consensus that the adoption of more formal API specifications might come with some well-known associated drawbacks (cost associated to difficulty and maintenance), which should be addressed by the proposed approaches.
- There is widespread interest on APIs specifying both functional and non-functional (quantitative) properties.

The above findings confirm the hypothesis in the work plan of the project, and provide additional inputs for the following tasks in WP 2. In particular, the development of models for behavioural APIs (T.2.1) needs to address the

description of behaviours and requirements stated in Chapter 3.1.4, i.e., they should be handled in the future development on the combinations of textual and declarative models for b-APIs and automata-based approaches to the modelling techniques.
In addition, the application domains of the partners in the consortium provide new challenges in the development of techniques for Static Verification (T.2.4). The current practices described in Chapter 3.2.3, can be complemented by behavioural types and contracts, for dealing with causal relationships, user authorisations and initialisation pre-conditions. Also, behavioural theories need to be extended with quantitative requirements identified in Chapter 3.4.3.
Also, partners have provided input for requirements tools and techniques for automatic documentation, testing of documentation and run-time monitor (as stated in Chapter 3.4.2). These requirements are aligned and serve as input for the following tasks: T.2.5 (Dynamic analysis) tailored to the development of techniques to use contracts to mechanically generate monitors to control interactions and non-functional properties; T.2.6 (Automatic test generation) to refine test generation approaches for contract-testing frameworks for languages and technologies described in Chapter 3.2.2.

Among the surveyed partners only one has a publicly available description of an API, whose information is provided in Chapter **3.1.3 Use cases of API provisioning.** Remaining partners are open to give access to their APIs under non-disclosure agreements to other partners of the project in order to be used as additional use cases for the remaining activities of the project. Depending on the nature of the agreements to be set between partners, it is expected that more detailed descriptions of the remaining use cases will be provided in future deliverables and publications of the project.

The results reported in this document have not been published elsewhere. However, we expect some of the conclusion of this activity to be communicated as part of a planned publication about the project and its preliminary results.

This deliverable will be available at the website of the project https://www.um.edu.mt/projects/behapi/ in due course.