



Project N°: **FP7-610582**  
Project Acronym: **ENVISAGE**  
Project Title: **Engineering Virtualized Services**  
Instrument: **Collaborative Project**  
Scheme: **Information & Communication Technologies**

## **Deliverable D5.7**

### **Business Strategy & Revised Exploitation Plan**

Date of document: M24



Start date of the project: **1st October 2013**

Duration: **36 months**

Organisation name of lead contractor for this deliverable: **ATB**

Final version

STREP Project supported by the 7th Framework Programme of the EC		
Dissemination level		
PU	Public	✓
PP	Restricted to other programme participants (including Commission Services)	
RE	Restricted to a group specified by the consortium (including Commission Services)	
CO	Confidential, only for members of the consortium (including Commission Services)	

# Executive Summary:

## Business Strategy & Revised Exploitation Plan

This document summarizes deliverable D5.7 of project FP7-610582 (**Envisage**), a Collaborative Project supported by the 7th Framework Programme of the EC, within the Information & Communication Technologies scheme. Full information on this project is available online at <http://www.envisage-project.eu>.

In this document we present the results of the business strategy and revised exploitation plan activities performed in the second year of the **Envisage** project. The previous related deliverable (D5.6) focused on market analysis and description of exploitable products developed in the **Envisage** project, whereas this document focuses on business strategy. This document has particular emphasis on two key areas: viable revenue streams and marketing channels. The exploitation activities reported in this document and its results need to be considered in their initial phase. Updates on the exploitation activities and results will be given in the following versions of the deliverable: D5.8 Standardization Activities & Final Exploitation.

## List of Authors

Amund Tveit (ATB)

Input from **Envisage** Consortium members through a Business Strategy Questionnaire

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Exploitables . . . . .	5
1.1.1	ABS Modeling Language and Toolset . . . . .	6
1.1.2	Virtual Collaboratory . . . . .	7
1.1.3	Code Generator . . . . .	7
1.1.4	Monitoring Add-Ons and Visualization . . . . .	7
1.1.5	Deadlock Analysis . . . . .	8
1.1.6	Test Case Generation Framework . . . . .	8
1.1.7	Deductive Verification Framework . . . . .	8
1.1.8	Resource Analysis Framework . . . . .	8
1.1.9	Simulation Tool . . . . .	8
1.2	Market Overview . . . . .	9
1.2.1	Cloud Computing Market . . . . .	9
<b>2</b>	<b>Business Strategy</b>	<b>11</b>
2.1	Value Proposition and Success Factors . . . . .	11
2.2	Revenue streams . . . . .	11
2.2.1	Combination of Open Source and Licensed Software . . . . .	15
2.2.2	Professional Service for Open Source Software . . . . .	15
2.2.3	Full Service: lifetime responsibility for customer's software . . . . .	16
2.2.4	Cloud Service . . . . .	16
2.2.5	App Store Model . . . . .	16
2.2.6	Training and Certification . . . . .	16
2.2.7	Commercial Conferences . . . . .	16
2.3	Marketing and Sales Channels . . . . .	17
2.3.1	Viral marketing . . . . .	17
2.3.2	Public Relations (PR) . . . . .	17
2.3.3	Unconventional PR . . . . .	17
2.3.4	Search Engine Marketing (SEM) . . . . .	17
2.3.5	Social & Display Ads . . . . .	18
2.3.6	Offline Ads . . . . .	18
2.3.7	Search Engine Optimization (SEO) . . . . .	18
2.3.8	Content Marketing . . . . .	18
2.3.9	Email Marketing . . . . .	18
2.3.10	Engineering As Marketing . . . . .	18
2.3.11	Targeting Blogs . . . . .	19
2.3.12	Business Development (BD) . . . . .	19
2.3.13	Sales . . . . .	19
2.3.14	Affiliate Programs . . . . .	19
2.3.15	Existing Platforms . . . . .	20

2.3.16	Trade Shows . . . . .	20
2.3.17	Offline Events . . . . .	20
2.3.18	Speaking Engagements . . . . .	20
2.3.19	Community Building . . . . .	20
2.4	Major Risks to be Assessed . . . . .	20
2.4.1	Product-Market Fit . . . . .	20
2.4.2	Investments Required . . . . .	21
2.4.3	Leadership and Talent Required . . . . .	21
2.4.4	What about the competition? . . . . .	21
<b>Bibliography</b>		<b>21</b>

# Chapter 1

## Introduction

In this document we present the results of the business strategy and revised exploitation plan activities performed in the second year of the **Envisage** project. The previous related deliverable (D5.6) focused on market analysis and description of exploitable products developed in the **Envisage** project, whereas this document focuses on business strategy. This document has particular emphasis on two key areas: viable revenue streams (Section 2.4) and marketing channels (Section 2.5). The exploitation activities reported in this document and its results need to be considered in their initial phase. Updates on the exploitation activities and results will be given in the following versions of the deliverable: D5.8 Standardization Activities & Final Exploitation.

This chapter first describes the exploitable products and features developed in the **Envisage** project, followed by summary figures of market characteristics.

### 1.1 Exploitables

This section provides the exploitables (**Envisage** products and features) originally described in Section 2.1.2 in deliverable D5.6 and added in Section 1.1 for completeness. They are categorized in Table 1.1 and connected to value propositions (i.e., customer problems and **Envisage**'s solution to them) in Table 2.3.

**Envisage** is in general about modeling and detecting quality issues early, Figure 1.1 shows the importance of this, it can cost approximately 100 times more to correct a defect in the (late) maintenance stage rather than in the (early) requirement stage [2]. This is an underlying motivation for all exploitables.

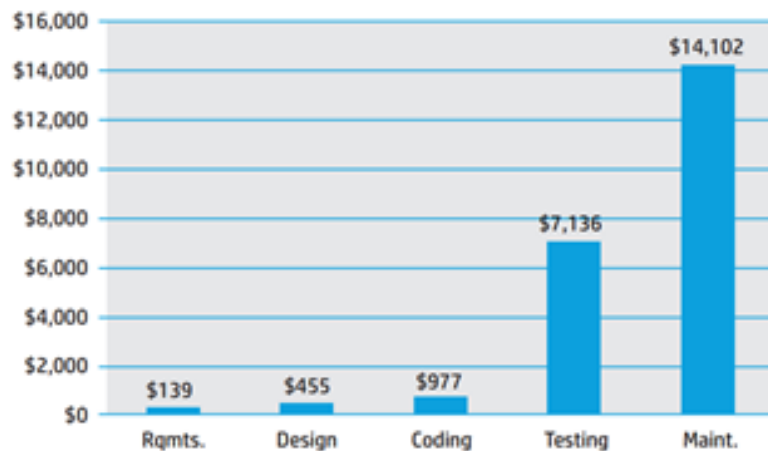


Figure 1.1: The costs of correcting defects throughout the software lifecycle

#	Category	Name
P1	System Modeling & Software Engineering	ABS Modeling Language and Toolset
P2	System Modeling & Software Engineering	Virtual Collaboratory
P3	System Modeling & Software Engineering	Code Generator
P4	System Modeling & Software Engineering	Deadlock Analysis
P5	System Modeling & Software Engineering	Test Case Generation Framework
P6	Cloud Service Quality	Monitoring Add-Ons and Visualization
P7	Cloud Service Quality	Deductive Verification Framework
P8	Cloud Resource Analysis	Resource Analysis Framework
P9	Cloud Resource Analysis	Simulation Tool

Table 1.1: Envisage Exploitable Feature Offerings (EEFO) (See also previous deliverable D5.5)

### 1.1.1 ABS Modeling Language and Toolset

**Envisage** is based on the modeling language ABS (Abstract Behavioral Specification) [8]. Here we briefly describe the ABS eco system and the development prospects of ABS. The ABS language is the main outcome of the FPT FET Integrated Project — Highly Adaptable and Trustworthy Software using Formal Models (HATS<sup>1</sup>) — that ran from April 2009 until March 2013. The ABS eco system consists of two main parts: the ABS modeling language and the ABS tool set which we describe now briefly in turn.

**The ABS Modeling Language.** The ABS modeling language is a fully executable, concurrent language whose basis is a modern object-oriented/functional design paradigm. ABS has a formal semantics and is extensively documented in [1]. There is also a tutorial on ABS [5]. Compared to other modeling languages, ABS has a number of distinctive features:

- ABS permits automatic code generation. Currently supported target languages include Java, Erlang, and Haskell.
- ABS includes sublanguages for feature description and to model product variability. Specifically, it is possible to define product lines in ABS [6]. Together with code generation, this makes end-to-end modeling of product lines possible.
- ABS can naturally capture the properties of a runtime environment [9]. Deployment components allow the developer to access and process at the modeling level the resource parameters of the machine(s) where a service is running. This is crucial for the intended usage of ABS within **Envisage**.
- ABS has been developed in tandem with a number of state-of-art software productivity tools to assure their scalability (see next section).

ABS is well established in the academic formal methods and programming languages communities. For example, the standard ABS reference [8] received well over 100 citations in Google Scholar in the three years since its publication. Tutorials about ABS have been presented at major international conferences, such as ECOOP and FM. A summer school [3] and a track of an international conference [7] were to a large extent devoted to ABS. A dedicated website for ABS exists that is continuously developed.

**The ABS Tool Set.** ABS comes with a number of mature and sophisticated tools [11] that go well beyond what is available in commercial IDEs. The ABS tool set contains simulators, type checkers, code generators (for Java, Haskell, Erlang), glass box test generation, complete deadlock and livelock analysis, resource analysis, formal verification. Several of these tools are described in the tutorial [4].

<sup>1</sup>The HATS project web page: <http://www.hats-project.eu>

### 1.1.2 Virtual Collaboratory

The **Envisage** virtual collaboratory is a (virtual) place where tools developed in the context of **Envisage** are made available to be used in different forms, e.g., as services, through a web-interface, by downloading to use locally, etc. Moreover, it allows users to share their experience and provide feedback. To facilitate the integration of **Envisage** tools in the virtual collaboratory, we will develop a methodology in which tools output their results in some predefined text-based language which is then interpreted by several interfaces, also developed within **Envisage**, and the results are shown graphically in several development environments (e.g., web-interface, Eclipse plugin). The technology developed will be used by programmers in the **Envisage** project to easily integrate their tools in the collaboratory. The collaboratory itself will be then available for internal and external users in order to easily experiment with the developed tools. The collaboratory is further described in deliverable D5.2.1.

### 1.1.3 Code Generator

For the generation of production code from ABS models two backends are developed: One backend for generating Java code and one for generating Haskell code. The modularity and internal APIs of ABS supports developing additional backends if needed, e.g. by a large customer that has invested heavily in a particular language.

### 1.1.4 Monitoring Add-Ons and Visualization

**A generic model for data point expressions.** We develop a formal methods approach to generalize how data points in a monitoring system can be expressed. A generic implementation model/API is delivered to enable definition, measurement, exposure and integration of data points. For example, a data point can be defined to extract QPS (query per second) from the application logs and expose it as a data point in the monitoring system using the monitoring API. Or, another data point can be defined to measure and expose the data characteristics of a customer such as data categories or data API usage.

**A systematic process and methodology to generate monitoring system components and plugins.**

1. Develop a methodology that enables a business stakeholder to initiate a model from a customer's contract and capture the requirements with the operations of the monitoring system.
2. Automate a process in which a customer's contract can be validated and then used to generate a model. From the model, parts of the monitoring system are generated that contribute to maintaining QoS of the service based on the customer's contract.
3. Evolve the service as necessary based on the monitoring measurements over the data points and metrics defined.

When a customer contract is being negotiated, the model above allows to validate the contract against the operational aspects of the offered service. For example, if the customer's contract has a notion of QPS as a QoS metric, the model can be used to validate the offering and make tuning if necessary. In the end, the contract terms are transformed into the model and then monitoring system components for the contract are generated to monitor the service for the customer and ensure the QoS if necessary.

The approach enables diverse personas to use metrics and KPIs: A diverse range of personas in a business organization demand different outcomes of a monitoring system. The monitoring data model allows a persona to define the data points as expressions in the monitoring system from the point of view of their interests. The monitoring system facilitates the persona to visualize the measurement of the defined data points. Thus, it helps to make an understanding of the service according to the contract. This helps the persona to build a metric or a KPI that allows business decisions.

In the above example, one can build a notion of QoS over the measured QPS data points defined in the monitoring system. This can be a basis to define capacity/availability of the service that is documented in the customer's contract.

### 1.1.5 Deadlock Analysis

The **Envisage** Deadlock Analysis tool will provide a set of tools to detect livelock and deadlock in (multi-threaded) concurrent code. Bugs due to concurrency are among the hardest to catch with traditional testing since timing plays a part and can make reproducibility very hard without dedicated tools for it.

### 1.1.6 Test Case Generation Framework

The **Envisage** Test Case Generation Framework will provide a set of test cases which guarantee that the selected coverage criterion is achieved. It adapts the aPET test case generation framework to **Envisage** needs. The framework receives as input an ABS program, a selection of methods to be tested, and a set of parameters that include a selection of a coverage criterion. More in detail aPET will be integrated within the ABS Eclipse IDE, and also within the **Envisage** virtual collaboratory. The generated test cases will be displayed in textual mode and, besides, it will be possible to automatically generate ABS Unit test cases. The information yield by aPET can be relevant to spot bugs during program development and also to perform regression testing.

### 1.1.7 Deductive Verification Framework

The **Envisage** Deductive Verification Framework will extend KeY-ABS, a deductive verification system for ABS. KeY-ABS is based on the verification system KeY which verifies sequential Java programs and a variant of dynamic logic called ABS Dynamic Logic. KeY-ABS deals with concurrent ABS programs and uses history-based functional specification of ABS models in terms of method contracts, class invariants and interface invariants. With the Deductive Verification Framework we intend to extend the current history-based specification system to specify SLAs in terms of the restriction between resource consumption, resource provision and load balancing. Using the developed framework it will be possible to formally prove that a system modeled in ABS is able to guarantee certain resource and load balancing properties that are part of an SLA. The used logic and verification method is compositional and modular. This means it does not make any additional assumptions on the number of objects, cogs or similar (except those specifically mentioned as part of an SLA).

### 1.1.8 Resource Analysis Framework

The **Envisage** resource analysis framework allows us to determine at an early stage of software development the resource consumption of abstract behavioral specification models (ABS). The information gathered focuses on concrete components of the system (like the number of executed instructions or memory consumption at each location) but also on system-level properties (like the number of spawned tasks or the amount of data transmitted among the locations of a virtual system). This will allow the users to anticipate potential bottlenecks in the locations of the system and to optimally distribute the load of work in order to fulfill the service contracts of the components services. Developers can use the resource analysis framework to check that the resource consumption of each component fulfills the service contracts, and system designers can use the framework's results to better deploy the system and avoid potential bottlenecks.

### 1.1.9 Simulation Tool

The **Envisage** Simulation Tool is a simulation environment for the abstract modeling language. It combines system level descriptions with resource and deployment models to allow rapid prototyping. The simulation environment will allow the rapid prototyping of models in different deployment scenarios and with different



load balancing strategies. The **Envisage** simulation environment will integrate state of the art tools for development of formal languages and execution platforms for operational semantics with compiler generation and type checking in an IDE with editing and visualization support (such as Eclipse).

## 1.2 Market Overview

This section provides highlights from the market analysis described in the previous deliverable D5.6.

### 1.2.1 Cloud Computing Market

The overall cloud computing market size and estimated growth is shown in Figure 1.2 (*these estimates come from IHS Technology*), followed by migrations to cloud market (from traditional data centers and server rooms) broken down by functional area in Figure 1.3 (*these estimates come from KPMG*).

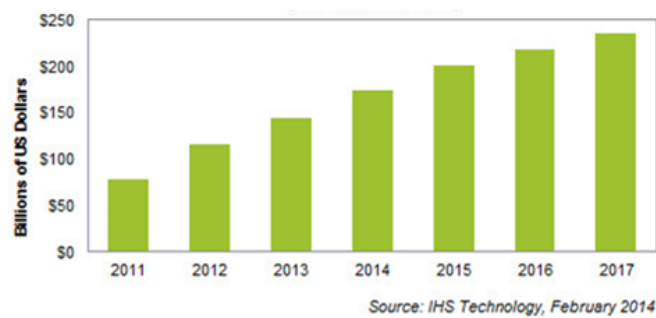


Figure 1.2: Business investments on cloud computing worldwide, IHS

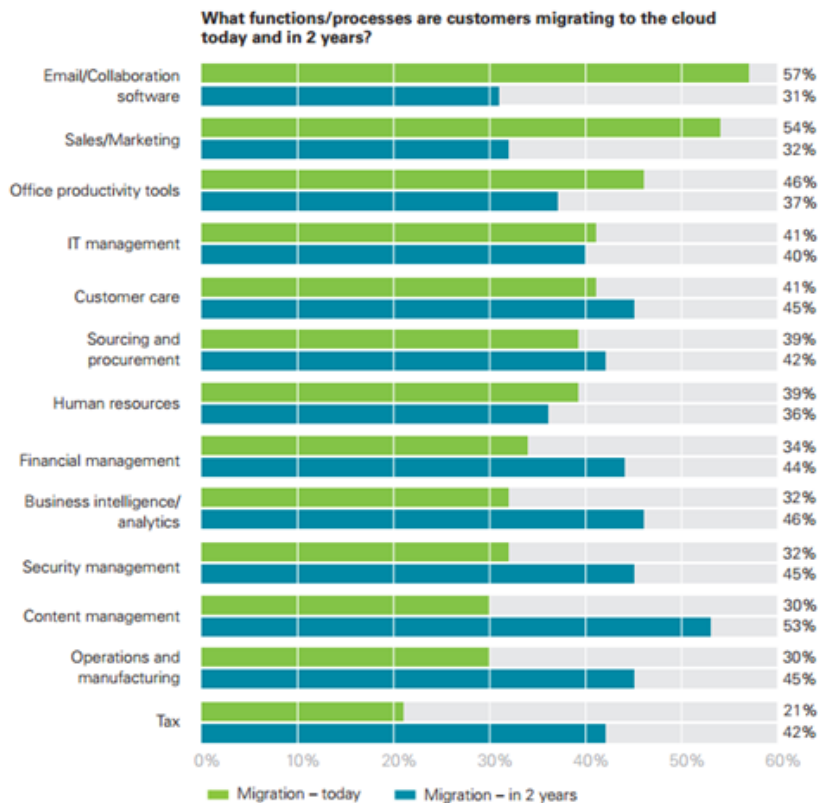


Figure 1.3: Functions migrated to cloud (Source: KPMG International's 2012 Global Cloud Providers Survey)

Two different estimates for how the Cloud Computing market is split between various types of cloud infrastructure (e.g., SaaS, IaaS, and PaaS) is shown in Figure 1.4 (*estimates from Forrester Research*) and 1.5 (*estimates from Gartner Group*).

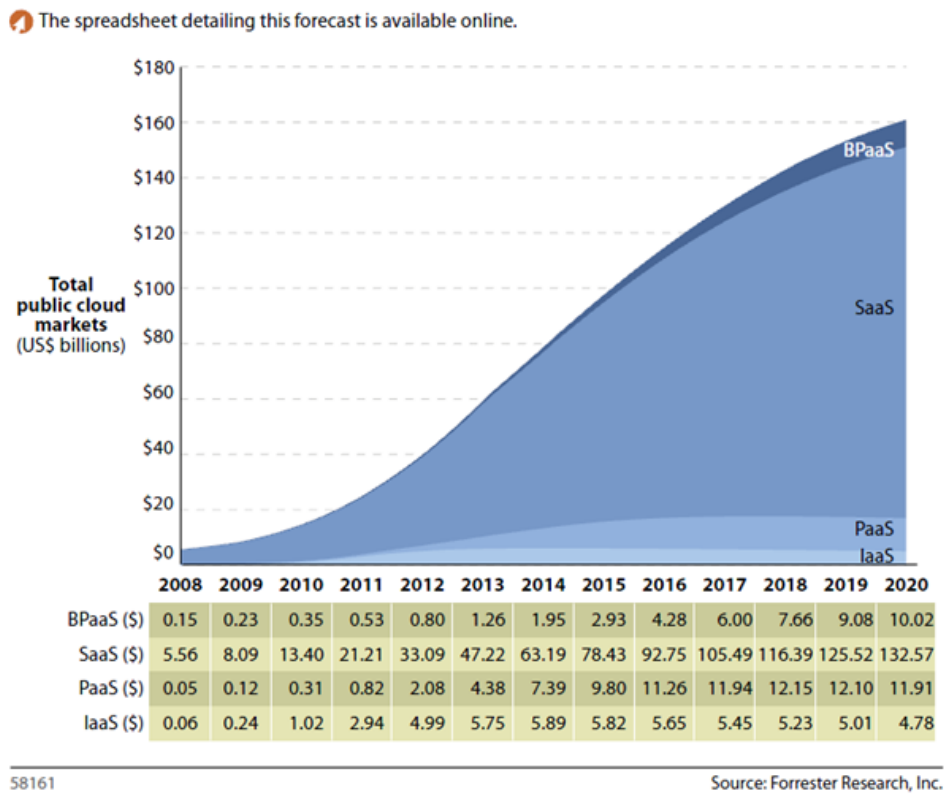


Figure 1.4: Global public Cloud market size 2011 to 2020, source Forrester Research

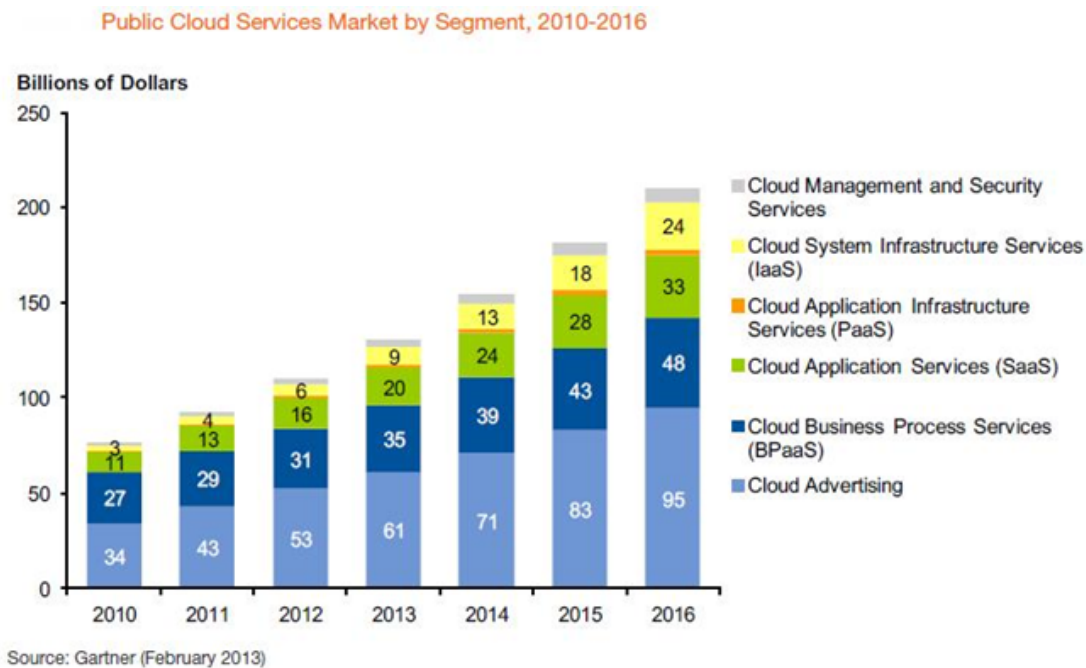


Figure 1.5: Public Cloud Services Market by segment, 2010–2016

## Chapter 2

# Business Strategy

This chapter provides the business strategy by describing:

1. High level value proposition
2. Critical success factors and how to measure those
3. Stakeholders, both existing adopters and potential adopters of **Envisage** technology
4. Marketing and sales channels
5. Key risk factors

In the next deliverable (D5.8), we plan a revisit of the market analysis, in particular the competitor analysis (since the market we operate in is rapidly changing), as well as having a more detailed plan with more information how to mitigate risks described in this deliverable.

### 2.1 Value Proposition and Success Factors

Table 2.3 presents the **Envisage** value proposition as a match between customer problems and the corresponding **Envisage** solution to each problem, and finally a relation between the **Envisage** solution and exploitable features (EEFO). In order for **Envisage** to succeed it needs to provide *real* benefits to customers, this is presented as a list of success factors and how to measure those in Table 2.4.

### 2.2 Revenue streams

Software-related revenue streams have traditionally (until recent years) been characterized by:

1. a one-time purchase cost (common in the consumer market), or
2. an annual or periodic license fee (common in the enterprise market), that could depend on many variables (e.g., number of active users, or total number of users, site license etc).

This is usually, except for very narrow niches, *not* likely to be a viable business model for **Envisage** Exploitable, of two main reasons:

1. Proprietary (Closed Source) Software is increasingly seen as a risk factor (compared to Open Source), e.g., wrt. to both cost, security and dependency on Software Vendor. Programming languages and related tools are increasingly becoming or are on their way to becoming *open source*, e.g., C# and .Net from Microsoft, Swift from Apple, Go from Google, and RUST from Mozilla.

Stakeholder	Type	Example Organizations	Main Goals (Incentives/Interest in Envisage)
Enterprise	Early Adopter	Deutsche Bahn AG (German Railways) and Swiss Railways (SBB)	DB AG and SBDD will use ABS and its tools to model, simulate and analyze train movements according to the legal regulations for German railway operations. The aim is to be able to evaluate whether certain modifications of the signaling rules can increase the throughput of a train network without sacrificing its safety. This project (separate from <b>Envisage</b> ) starts in fall 2015. It is an industrial project (fully financed by DB AG and SBB) where ABS technology developed during <b>Envisage</b> is used.
Government	Early Adopter	Hessian Ministry for Science and the Arts	In the project COMPUGENE financed by the research excellence initiative of the state of Hesse (Germany) we will use ABS to model and analyze biological circuits. The ABS code generation tools will be adapted to synthesize the biological compounds.
Academia	Early Adopter	Graz University of Technology, TU Wien, University of Indonesia, Inria	Use of <b>Envisage</b> ABS tools in education and research. Benchmarking and comparison of technologies. Publish best-practice guidelines
Open Source	Indirect	Python, Haskell, D, Scala, Go, Hadoop YARN, Apache Spark, Apache Lucene	Fix of default sorting algorithm - TimSort - caused by bug finding and evaluated fix by <b>Envisage</b>

Table 2.1: Stakeholder Groups and Goals - Existing Adopters

Stakeholder	Type	Example Organizations	Main Goals (Incentives/Interest in Envisage)
Enterprise	Pot. Adopter	Transportation, e.g., airports, airway companies, buses, railways, street cars	Probably several of the same challenges as German and Swiss Railways have
Enterprise	Pot. Adopter	E-Commerce and On-line Service Vendors	Resource modeling and simulation of both cloud server side and client side (similar to ATB case study) and automatic generation of monitors (similar to FRH case study). Software Quality assurances with deadlock analysis
SME	Pot. Adopter	Mobile App Publishing Companies and Developers	Resource modeling and simulation of both cloud server side and mobile app side (similar to ATB case study) and automatic generation of monitors (similar to FRH case study)
Government	Pot. Adopter	European Countries	Modeling and Simulation of Government-provided Cloud or Online Services. Automatic monitor creation for those services. ABS Modeling of tax rules. Can e.g. simulate effect of changes and also formally validate the developed model to check for inconsistencies. Lessons from the project with German Railways can perhaps be used as input to this
Software Provider	Pot. Adopter	Oracle, IBM, SAP, JetBrains	Integration of Envisage tools with their own offerings (e.g., a JetBrains IDE for Envisage ABS and tools)
Standardization	Interested	ISO, W3C	Standardization of tools and techniques developed in Envisage
Consulting	Promoter	Cap Gemini, Tieto Enator, Steria	Rapid modeling and simulation of services for their clients, and increase service quality. Sell training related to Envisage tools and practices

Table 2.2: Stakeholder Groups and Goals - Potential Adopters

#	Problem	Solution Offered by Envisage	EEFO #
VP1	<b>System Modeling &amp; Software Engineering.</b> With the increasing number of parallel CPU cores in computers, <i>concurrency</i> is the key to increase performance and efficiency. Unfortunately there are 2 related issues with concurrency: 1) it is hard to program in most programming languages and 2) bugs due to errors in concurrency is common and very hard to avoid	Envisage offers a concurrent language - ABS - that has analysis tools to efficiently detect problems (e.g., livelocks or deadlocks) with concurrent code. Since the ABS language has formal semantics it is easier to develop sophisticated code analysis tools to support software engineers. The Envisage Virtual Collaboratory makes it easy to integrate ABS tools and services in existing development workflow	P1-P5
VP2	<b>Cloud Service Quality.</b> Enterprises develop highly critical cloud software that people's lives may depend on, e.g., in transportation, energy and healthcare, but verifying the correctness of this software is hard. Problem: How to improve quality of cloud services?	Envisage makes verification of software easier with the ABS Modeling Language than traditional programming languages since it has formal semantics built in by design. Development can be done from any client since the IDE (Virtual Collaboratory) is accessible as a web application. ABS is extended with the KeY-ABS tool which supports deductive verification also for concurrent programs. The predecessor of KeY-ABS named KeY has been used by Envisage team to find a bug in the TimSort sorting algorithm which is the default sort algorithm in almost all major programming languages, KeY was also used to validate the proposed fix to the TimSort bug)	P1,P6,P7
VP3	<b>Cloud Resource Analysis.</b> Running and changing large-scale distributed systems or big data jobs in the cloud can be costly and hard to predict, since even minor software changes can significantly effect cloud resource costs. Problem: how to efficiently get good estimates of cloud resources?	Envisage offers a simulation environment for the ABS language. This simulation environment offers abstractions for common resources in the cloud, e.g., virtual machines and network capacity	P1,P8,P9

Table 2.3: Customer Problem and Corresponding Envisage Solution

#	Success Factor	How to measure effects
1	Reduction in Cloud System Modeling and Development Cost	Use evidence we have, e.g., the FRH case study (where ABS and Java development has been compared), and measure development time (and related metrics) and compare to historic projects. Perhaps ask Simula at UIO or an independent research organization to perform a large-scale comparison study
2	Reduction in Cloud System Maintenance and Running Costs	Perform benchmarks compared to other functionally comparable cloud systems
4	Access to people with knowledge about Envisage tools	Measure how many have received training in Envisage tools and at what level of competency they have reached
5	Access to Case Studies and Templates	Build demonstrators that can be generalized into templates, e.g., a mobile app connected to a cloud backend (ref ATB case study) or monitoring for an ecommerce service (ref FRH case study) and also publish customer case studies. <i>Note: publishing of customer case studies from an early stage is probably one of the reasons why Amazon Web Services (AWS) is a success</i>

Table 2.4: Success Factors

2. The old models of revenue streams are increasingly being replaced by Cloud Computing Services. These services provide software and (virtualized) hardware on several abstraction levels, e.g., Infrastructure as a Service (IaaS), Software as a Service (SaaS) and Platform as a Service (PaaS)

The traditional ways of software-based revenue might still work, but Envisage believes that newer models based on Open Source, Cloud Computing, and Mobile have higher sustainable potential, as described below.

### 2.2.1 Combination of Open Source and Licensed Software

Combinations of (free) Open Source code and Proprietary closed source software can be provided. Examples of companies who are doing this are Hadoop vendors MapR (with a high performance distributed file system) and Cloudera (with a Hadoop management tool), and Linux vendor RedHat. The proprietary part may have similar pricing models as traditional enterprise software.

### 2.2.2 Professional Service for Open Source Software

Supplemental Support and Services catering Open Source, with models that may resemble software licensing, but only covers support guarantees and not right to use software. Examples are Hadoop vendors Horton-Works, Cloudera, and Mapr. These vendors charge 2-3000 dollars per machine per year to provide support. A typical Hadoop cluster starts at perhaps 10-30 machines and can go upward to several thousand machines. Such companies typically invest heavily in supporting Open Source, so the sales pitch is in-depth know-how of the software. Examples from mobile apps include Realm.io that offers services around their open source mobile database.

### 2.2.3 Full Service: lifetime responsibility for customer's software

This revenue stream can be explained with a non-software example: Rolls Royce provides not only physical jet engines for aeroplanes, but also provides guarantees and maintenance. They get paid for delivered airmiles. This is a step further than providing support when needed, since it requires attendance at all times. Example from mobile software include AppAfterCare.com that maintains (e.g., monitors and fixes bugs, or adds new functionality) a mobile app through its entire lifetime. They can charge up to several thousand Euros per month per customer for this level of support.

### 2.2.4 Cloud Service

Cloud Services are typically priced per resource per time unit, the range is typically from a few cents to a few Euros per hour per virtual machine, and other similar models for Cloud Services not connected to individual virtual machines. The Envisage Collaboratory could be a candidate for that, perhaps combined with a freemium model, with some resources available for free (e.g., software engineering tools) and some for additional cost (e.g., advanced code analysis tools).

### 2.2.5 App Store Model

The App Store model is currently mainly used in the consumer mobile market, but it might be that the enterprise market is moving in this direction for mobile and tablet solutions (e.g., given the recent partnerships Apple has made with IBM and Cisco in order to strengthen enterprise offerings and adoption of apps).

### 2.2.6 Training and Certification

Commercial training and certification for the use of Envisage ABS tools might become a good revenue stream. Examples of relatively recent successful cases of such revenue streams are Certified Scrummaster Training and related courses (e.g. product owner) from the Agile Alliance, a mix of free and subscription iOS development tutorials from [raywenderlich.com](http://raywenderlich.com), and certifications related to the Big Data infrastructure Hadoop from Cloudera. Commercial Training and Certification might be a prerequisite when targeting larger enterprise customers. If those who are trained are satisfied with the course sold to those companies, it might become easier with newly trained internal champions of Envisage offerings. Such training could physically happen on the premises of the customer or periodically in cities with the largest and most potential market. Another, more scalable alternative is to offer online courses and certifications, e.g. provided through a service such as [Udemy.com](http://Udemy.com), [Skilljar.com](http://Skilljar.com), [Wistia.com](http://Wistia.com), [Schoolkeep.com](http://Schoolkeep.com), [Goruu.com](http://Goruu.com), or [Usefedora.com](http://Usefedora.com). The online learning services can simplify the payment and technical handling of courses, and some of the online learning services offer whitelabelling so the course can be held from the Envisage business domain name. Since some Envisage tools are of complex nature, it is likely that some training is required to be able to take advantage of them. This means that there might be significant revenue potential from training.

### 2.2.7 Commercial Conferences

When a critical mass of customers or other users of Envisage technology has been reached, one might consider arranging a commercial conference. In this conference there could be a mix of customer case studies, Envisage case studies and tutorials as well as exhibitions for Envisage partners. Such conferences, in addition to the revenue from the conference itself, can be used as a recruitment channel for new customers and in general to strengthen the Envisage commercial community (e.g., if customer A shares Envisage best practices and experiences with customer B, then Envisage could gain from that). Examples of successful commercial conferences are Salesforce's DreamForce conference in San Francisco with 135000 attendees, another example is the Strata + Hadoop World conference (originally initiated by Cloudera).



## 2.3 Marketing and Sales Channels

In order for an early stage business (i.e., **Envisage**) to become successful, it needs *traction*. Traction is defined in the book “Traction: A Startup Guide to Getting Customers” [10] as: *‘Traction is basically the quantitative evidence of customer demand’*. So if you’re in enterprise software, [initial traction] may be two or three early customers who are paying a bit; if you’re in consumer software the bar might be as high as hundreds of thousands of users”, in the same book they suggest 19 different channels that can increase traction, hence improve marketing and sales. Below, these channels are described from an **Envisage** business strategy perspective.

### 2.3.1 Viral marketing

Viral marketing is about growing the userbase by encouraging users (customers) to refer to other users. In a more indirect form this happens to some degree with **Envisage** ABS tools today; e.g., when researchers and students at universities publish research papers (e.g., Inria) or video demonstrations (e.g., University of Indonesia) where **Envisage** tools are being used.

One strategy to encourage viral marketing could be to make it easy to share information about **Envisage** tools to social media and email directly from tools themselves, and perhaps also to allow invites by integrating with individual customer’s address book. Email-based invites directly to documents or spreadsheets was one of the reasons why Google for Work (*past name: Google Docs & Spreadsheets*) grew very fast. Allowing automatic sharing within same domain can also be a viable viral marketing strategy; e.g., if person@foo.com signs up and creates an ABS model, then everybody at foo.com can access it.

### 2.3.2 Public Relations (PR)

Publication Relations is about getting the **Envisage** business message published as non-advertisement in traditional media channels like newspapers, magazines, and perhaps also TV. In the simplest form it can be a press release published to a press release aggregator such as PR Newswire, where a press release might be found and quoted in newspapers. Broader research magazines such as IEEE Computer and Communications of the ACM can also be seen as part of PR.

With new and rapidly growing publishing platforms such as medium.com, developed by Evan Williams (the founder of both Blogger and Twitter), the difference between press releases, blog posts, and other article types is starting to blur. Many articles are in the category of infomercial. In addition to try to get the attention directly from relevant journalists, it can make sense for **Envisage** to publish the same or similar messages across PR channels, e.g., publish both on medium.com and PR newswire.

### 2.3.3 Unconventional PR

Unconventional PR is doing something out of the ordinary in order to get PR, but this is probably not a viable strategy for **Envisage** since this type of PR is better suited for the consumer market. But, if the story is powerful enough (hypothetically: one used formal methods in **Envisage** to prove exploits or errors in commonly used encryption algorithms) one could perhaps consider doing an unconventional PR stunt.

### 2.3.4 Search Engine Marketing (SEM)

Search engine marketing is about intention-based marketing, i.e. the web search queries and advertisements are shown together to the potential customer. For example,, an appropriate advertisement served together with the web search query “formal methods” could be an advertisement for **Envisage**. One potential source of relevant web search queries could be referrer web search queries (e.g., from Google) in web server log for the current **Envisage** blog (i.e., how people found the blog using search), in order to get more visits for the same keywords.

### 2.3.5 Social & Display Ads

Social ads on, e.g., Twitter, LinkedIn and Facebook support the accurate targeting on people; e.g., if one wants to serve ads to people working for a certain company or set of companies in a geographic region (city-level accuracy), that is possible. There are also opportunities to target people based on keywords on people's profile or on their work title (e.g., software engineer).

These types of ads are less transactional by nature than search (intent) ads, but can be good to create awareness and also for market analysis. An example of the latter could be to run ads with a certain **Envisage** message (e.g., cloud service quality) towards a potential customer group carefully targeted. This can be a lightweight way to get some indications of interest from a customer group, and can be compared to other **Envisage** ad campaigns to know how well it performs.

### 2.3.6 Offline Ads

Offline ads, such as TV spots, radio commercials, billboards etc. is probably not an efficient marketing channel for **Envisage** at an early stage.

### 2.3.7 Search Engine Optimization (SEO)

A simple strategy for SEO is to make all published content about **Envisage** (e.g., web pages) available so it renders well and loads fast on both small (mobile), medium (tablet) and large screens (laptops and desktops). Publishing links to produced content in appropriate and relevant fora (e.g., **Envisage**'s Twitter and Facebook feed) so it gets discovered faster might also help.

### 2.3.8 Content Marketing

Content marketing in the form of blogging has been very efficient in **Envisage** project so far, with the blog post about the TimSort bug as a highlight. In order for blogging to be efficient over time, one needs to post regular updates to the blog, e.g., once a week or more often. One typical danger when blogging is to be too self-critical about the quality of blog posts. This typically slows blog post publishing frequency a lot. It is important to be aware that blog posts are a lightweight and rapid communication channel that due to its rapidness differs significantly in quality compared to camera-ready academic papers.

### 2.3.9 Email Marketing

Email marketing is considered the second most effective online marketing channel after search engine marketing, but one has to be careful how to use it since many people have a hard time discerning email marketing and spam (and who can blame them?). A good approach for **Envisage** can be to start gathering email addresses from users by adding highly visible signup forms on web pages and on blog posts, and start experimenting by sending out emails periodically with new blog posts, case studies, and finalize with a call to action (e.g., surveys, sign up for new products, etc.)

### 2.3.10 Engineering As Marketing

Engineering as marketing can be, e.g., the creation of micro-sites that have **Envisage** product demonstrators, widgets for integration with other tools, or minor tools that support the main **Envisage** products. This combined with content marketing (blogging) and email marketing (in particular forms to gather customer leads) can be a good marketing channel. A good place to start here could be in the form of academic case studies which can be turned into specific blogs and tools addressing specific markets (e.g., model-based configuration support for Hadoop YARN clusters).

### 2.3.11 Targeting Blogs

Targeting blogs is about finding large and relevant influencers in popular online communities, e.g., if **Envisage** was publishing games and the game was tested by PewDiePie - the largest YouTube star - this could have a massive effect on marketing. In the **Envisage** project's popular TimSort blog post, we got a flavor of this by being mentioned on Twitter and Facebook by prominent members of the global programming community. In order to take full advantage of this for **Envisage**, a more quantitative approach should be taken:

1. Find and prioritize relevant online communities (e.g., YouTube, StackOverflow, Reddit, blogs, Product-Hunt, etc.)
2. Find and prioritize relevant and large influencers in those communities
3. Develop strategies for how to influence the selected influencers

### 2.3.12 Business Development (BD)

Business development is in some ways similar to targeting blogs, it is about who to partner with (typically larger companies or organizations) and how to approach them. But even more important is what to get out of the partnership. It can range from very lightweight partnerships (e.g., mention as being partner of big company X, which typically means very little but may add a tiny value of trust that can influence potential customers) to tighter partnerships (e.g., related to sales channels). In the beginning for **Envisage**, it is probably of greater value to have good customer referrals. For example, having customer testimonials from highly trusted organizations such as Swiss and German Rails would be of great value to have on a web site. An ideal customer testimonial would be someone at a (relatively) high level in Swiss or German Rails saying something nice about the **Envisage** technology that can be used (together with an image that person and the logo of the company) on the front web page of **Envisage**.

### 2.3.13 Sales

The first customers of **Envisage** will help shape the products. When the products are clearly shaped and one has a few customer referrals, it makes sense to start building up a sales force. Engaging sellers too early can be a risk, since they don't have a clear product to sell. After a few initial customers (where the founders of **Envisage** sell the products themselves), this is more likely to change. However, preparation for sales can happen earlier; e.g., by having a Customer Relationship Management system that integrates some of the other marketing channels (e.g., email marketing) and bug tracking systems. Prior to building up sales, it will be useful to hire a product manager in order to improve product descriptions, product-related metrics, the product roadmap, and function as a communication layer between sales and development team.

### 2.3.14 Affiliate Programs

Affiliate programs is when a third party web site, mobile app or similar links to a product and gets a percentage of the revenue if a click from that web site leads to a sale. Amazon.com is the biggest e-commerce site in the world and has a significant percentage of their revenue through affiliate sales through clicks from 3rd party web sites (there is an unconfirmed number of 40% of revenue).

Affiliate programs are used even more aggressively for products that have close to zero marginal cost; i.e., products that can be electronically distributed. Examples of such products are software, e-books, and online learning material. **Envisage** may provide both software and online learning. Since marginal cost is close to zero the percentages given to affiliate sales can be generous, in some cases more than 50%. Affiliate programs can be combined with targeting blogs in order to get influencers to get a fair cut of revenue when they mention **Envisage** products.

### 2.3.15 Existing Platforms

The integration of Envisage tools with existing platforms has already started with the Eclipse IDE plugins. However, Eclipse only serves a fraction of the developers. Making sure that Envisage technologies and ABS can be integrated with and used from other IDEs can improve uptake considerably, such as Microsoft Visual Studio, Apple's XCode, Google's Android Studio, JetBrains's IntelliJ and others. Integrating authentication with existing IDEs that are frequently used by developers can also help on uptake (i.e., less registration friction needed to get started with Envisage), such as Google for Work, Facebook, Twitter, Slack.com, Github, or Atlassian's Bitbucket. Since integration work might be costly, a similar quantitative prioritization approach as for targeting blogs above is suggested.

### 2.3.16 Trade Shows

Trade shows have traditionally been a good way to market software, and perhaps even more important: finding and getting deals with partners. Since Envisage is at an early stage of development, participating at trade shows should probably be delayed. However, one exception could be to participate together with a larger customer. Hypothetically: If, for example, Swiss or German Rail or Fredhopper were to participate to a large trade show and Envisage could participate within the area of one of those companies with the case study for that customer.

### 2.3.17 Offline Events

Offline events, such as large conferences (e.g., DreamForce mentioned earlier) and trade shows (without participating actively with a booth as previous section), can be useful for networking with potential customers and partners as well as learning about trends and competitors.

### 2.3.18 Speaking Engagements

Envisage team members are already doing speaking engagements, e.g., related to the detection of the TimSort bug. Speaking engagements can have a lasting effect after the speak itself since videos and slides are typically made available online. It can be a good idea to publish the slides on the Envisage web site and/or an Envisage Slideshare account, so analytics can be used to determine which organizations are interested in the material.

Speaking engagements at large and prominent conferences can also lead to interesting contact with influencers; e.g., speaker dinners with keynote speakers that might be high-ranked in large potentially interesting companies.

### 2.3.19 Community Building

Since Envisage is likely to take a partial or full open source based business model, a lot of the community building is going to be around Envisage's Github and development mailing lists. In addition to that, it can make sense to create community meet-ups when a critical mass of customers has been reached, perhaps as part of other larger events.

## 2.4 Major Risks to be Assessed

Starting up a business involves a diverse range of risks that needs to be mitigated, here are the main ones:

### 2.4.1 Product-Market Fit

Are we solving a real need? Do we know who the ideal customer is? What are the barriers to adoption of Envisage technology?

### **2.4.2 Investments Required**

What happens when project funding runs out? What kind of funding will be needed at that point?

### **2.4.3 Leadership and Talent Required**

What happens at the end of the project with leadership and talent (e.g., developers and marketers) to commercialize Envisage technology?

### **2.4.4 What about the competition?**

We have ideas about the competitors, but it is a rapidly changing market, so what was true earlier in the project isn't necessarily true when the project ends.

# Bibliography

- [1] *The ABS Language Specification*, 1.2.0 edition, April 2013. <http://tools.hats-project.eu/download/absrefmanual.pdf>.
- [2] Elvira Albert, Frank de Boer, Reiner Hähnle, Einar Broch Johnsen, and Cosimo Laneve. Engineering virtualized services. In M. Ali Babar and Marlon Dumas, editors, *2nd Nordic Symposium on Cloud Computing & Internet Technologies (NordiCloud'13)*, pages 59–63. ACM Press, 2013.
- [3] Marco Bernardo, Ferruccio Damiani, Reiner Hähnle, Einar Broch Johnsen, and Ina Schaefer, editors. *Executable Software Models: 14th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, Bertinoro, Italy*, volume 8483 of *Lecture Notes in Computer Science*. Springer-Verlag, June 2014.
- [4] Richard Bubel, Antonio Flores Montoya, and Reiner Hähnle. Analysis of executable software models. In Bernardo et al. [3], pages 1–27.
- [5] Reiner Hähnle. The Abstract Behavioral Specification language: A tutorial introduction. In Marcello Bonsangue, Frank de Boer, Elena Giachino, and Reiner Hähnle, editors, *International School on Formal Models for Components and Objects: Post Proceedings*, volume 7866 of *Lecture Notes in Computer Science*, pages 1–37. Springer-Verlag, 2013.
- [6] Reiner Hähnle, Michiel Helvensteijn, Einar Broch Johnsen, Michael Lienhardt, Davide Sangiorgi, Ina Schaefer, and Peter Y. H. Wong. HATS abstract behavioral specification: the architectural view. In Bernhard Beckert, Ferruccio Damiani, Frank de Boer, and Marcello M. Bonsangue, editors, *Proc. 10th International Symposium on Formal Methods for Components and Objects (FMCO 2011), Torino, Italy*, volume 7542 of *Lecture Notes in Computer Science*, pages 109–132. Springer-Verlag, 2013.
- [7] Reiner Hähnle and Ina Schaefer. Adaptable and evolving software for eternal systems — (track summary). In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change — 5th International Symposium, ISoLA 2012, Heraklion, Crete, Greece*, volume 7609 of *Lecture Notes in Computer Science*, pages 1–3. Springer, October 2012.
- [8] Einar Broch Johnsen, Reiner Hähnle, Jan Schäfer, Rudolf Schlatte, and Martin Steffen. ABS: A core language for abstract behavioral specification. In Bernhard Aichernig, Frank S. de Boer, and Marcello M. Bonsangue, editors, *Proc. 9th International Symposium on Formal Methods for Components and Objects (FMCO 2010)*, volume 6957 of *Lecture Notes in Computer Science*, pages 142–164. Springer-Verlag, 2011.
- [9] Einar Broch Johnsen, Rudolf Schlatte, and S. Lizeth Tapia Tarifa. Integrating deployment architectures and resource consumption in timed object-oriented models. *Journal of Logical and Algebraic Methods in Programming*, 84(1):67–91, 2015.
- [10] Justin Mares and Gabriel Weinberg. *Traction: A Startup Guide to Getting Customers*. S-curves Publishing, 2014.

- [11] Peter Y. H. Wong, Elvira Albert, Radu Muschevici, José Proença, Jan Schäfer, and Rudolf Schlatte. The ABS tool suite: modelling, executing and analysing distributed adaptable object-oriented systems. *Journal on Software Tools for Technology Transfer*, 14(5):567–588, 2012.