axini
testautomatisering

Afstuderen bij Axini

Houd je van breinkrakers, onderzoek vertalen naar de praktijk?
Ben je niet bang voor techniek, software development en toolontwikkeling?
Heb je zin om te werken in een klein, gespecialiseerd team?

Axini is een spin-off van de Universiteit Twente. Technologie en theorie die vers uit de onderzoeks-wereld komen passen we toe op complexe systemen van klanten. Daarbij ontwikkelen we onze eigen toolset.

Wat zijn jullie voor bedrijf?

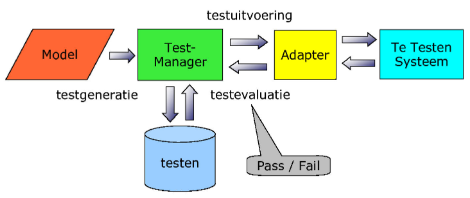
Axini is voortgekomen uit promotie-onderzoek naar model-gebaseerd testen. We hebben dit doorontwikkeld tot tooling waarmee de meest complexe systemen volledig automatisch en grondiger dan ooit kunnen worden getest.

Dit is uniek en erg succesvol. We hebben voorname klanten: banken, verzekeraars, logistieke partijen, fabrikanten van navigatiesystemen, betaalsystemen tot medische.

We groeien daarom flink maar zijn, met een tiental enthousiaste techneuten, nog prettig klein en informeel.

Wat is model-gebaseerd testen?

Testen van software bestaat uit drie stappen: bedenken van testen, uitvoeren van testen en het controleren van de testuitkomst. Model-gebaseerd testen kan deze drie

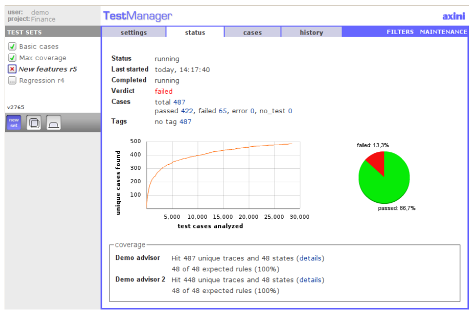


stappen automatiseren.

De functionaliteit van het te testen systeem wordt beschreven in een van onze domeinspecifieke modelleer-talen. Uit dit model kan TestManager, ons modelleer en test-tool, gericht testen afleiden, uitvoeren op het te testen systeem en vervolgens controleren. Daarna volgt analyse: hoe goed is er nu getest? Wat zijn mogelijk onderliggende foutoorzaken?

Waarom zou ik bij jullie afstuderen?

We hebben legio onderwerpen waar je relevant onderzoek naar kan doen. Van theoretisch tot praktisch, van constraint solving tot datavisualisatie, met veel ruimte voor eigen invulling.



Status running
Last started today, 14:17:40
Completed running
Verdict failed
Cases total 487
passed 422, failed 65, error 0, no_test 0
Tags no tag 487

Demo advisor Hit 487 unique traces and 48 states (details)
48 of 48 expected rules (100%)
Demo advisor 2 Hit 448 unique traces and 48 states (details)
48 of 48 expected rules (100%)

We bieden goede, inhoudelijke begeleiding en vaak is er de mogelijkheid om je werk in de praktijk te toetsen bij een van onze klanten. We werken met allerhande hippe open source spullen.

Er valt dus het nodige te beleven en te leren. En last but not least: ons kantoor is in Amsterdam, tussen de bakker en de kroeg.

Welke studenten zoeken jullie?

We zoeken slimme, gedreven studenten die van hun vak houden en er plezier in hebben om de gebaande paden te verlaten en nieuwe wegen te ontdekken. Veranderen van taal of tools deert je niet en je haalt voldoening uit het leveren van mooi werk.

Contact

Je vindt ons op Azartplein 12, Amsterdam. Bel voor meer informatie Menno Jonkers op (06) 2476 4265, of mail info@axini.com

axini
testautomatisering

Examples of potential graduation projects

In addition to the examples listed here, there are many other options in our broad field of research. Tell us what you're looking for and we can check if there's a fit.

1. Test selection strategy with data

Testing software is inherently incomplete; no matter how intensely a product has been tested, there remains some probability that one or more faults are present. Still, observing correct behavior increases our confidence in the correctness of a system. To quantify this increase of confidence, several notions of 'test coverage' and 'risk' have been developed. However, many of these notions are very basic, and therefore do not predict the quality of a test process in a very accurate manner.

Axini develops Axini TestManager. TestManager is a model-based testing tool that incorporates well-known concepts such as 'state' and 'transition' coverage. The models used for testing, besides having states and transitions, also contain data, constraints and formulae on them. TestManager has test selection strategies to record and maximize state and transition coverage. However we believe these strategies can be improved by extending the consideration of data. This results in an improved coverage strategy with data.

The purpose of this assignment is to research concepts of test coverage, specifically regarding data coverage, and implement a test selection strategy which maximizes the data coverage of the executed test.

We expect as deliverables a improved data coverage strategy and some practical application in Axini's domain with a proof of concept, preferable a case study or prototype tool.

Assignment contact: Vincent de Bruijn

2. Formal DSLs

Problem

The Ruby programming language allows easy implementation of domain-specific languages (DSLs), that can always fall back on the power of the full language to perform operations that cannot (easily) be captured by a DSL. Unfortunately, the model-based testing theory is usually based on state machines or transition systems and as such that theory does not apply to models written in embedded Ruby DSLs. For expressing their models, DSLs are preferred by our customers. As rigorous testing requires the theory to be applicable to the models, a way to bring these two together is required.

Note that a lot of research and knowledge about DSLs is available at the UvA, more specifically at the CWI SWAT group of Prof. Dr. Jurgen Vinju.

To be researched

Is there a transformation or morphism that describes a correspondence between a Ruby DSL and a symbolic transition system (our formalism of choice)? There are tools to construct the Abstract Syntax Tree of a piece of Ruby code, which resembles a series of state machines that operate on a set of overlapping data stores. Can this notion be formalized into a symbolic transition system?

Evaluation

Tools from Model-based testing theory can be used to guide test case generation by Ruby models. The amount of test coverage achieved can be expressed in terms of states and transitions.

3. Generic adapters

Axini develops, applies and sells model based testing tools. Model based testing (MBT) is a new and advanced approach to fully automatically test systems. Based on models (formal descriptions of system behavior) our tools simultaneously generate and execute tests. To support this approach a system under test should be connected to our tools. This is basically what the adapter component does: it bridges the gap between the model world and the actual system under test.

So every MBT application needs an adapter. Some communicate with web services, others with message brokers or some proprietary protocol over IP. Designing and building/configuring an adapter can take quite some effort. However in many different applications we see an overlap in functionality. This assignment addresses the analysis of adapters and design of a component framework to build and/or configure adapters. We aim here at reuse. The major goal is obtain (partial) answers to some of these questions:

- What characterizes an adapter?
- What is a suitable abstract formal model of an adapter?
- How should actions in a model correspond to real interaction with the system under test?
- Can we distinguish domain specific adapters?
- Out what components should an adapter exist?

We expect as deliverables a literature study about the state of the art, and some practical application in Axini's domain with a proof of concept, preferable a case study or prototype tool.

Assignment contact: Rene de Vries

4. Data mining and clustering of large test sets

Axini develops, applies and sells model based testing tools. Model based testing (MBT) is a new and advanced approach to fully automatically test systems. Based on models (formal descriptions of system behavior) our tools simultaneously generate and execute tests. One of the advantages over existing test approaches is that MBT can do large volume tests: a lot more tests in same amount of time. This gives more insight in the quality of the system under test.

Problem

Running a test on an application results in a sequence of stimuli (actions of the test tool) and responses (reaction of the system under test). A test case also has a verdict: whether it passed or failed, i.e. if all the responses were allowed according to the model of the system.

Fully automated testing with MBT often results in tens of thousands of these test cases. Many over them have (partly) similar sequences, correlating to (root) causes of errors. Humans can typically quickly detect such patterns and determine what test cases are probably due to a single underlying error, or what a minimal sequence would be to reproduce the error. But human analysis becomes unfeasible given tens of thousands test cases.

Goal

Look into sequential pattern mining and cluster analysis techniques that can help us to:

- group ‘similar’ test cases (consult domain experts to define similarity)
- select a limited number of cases that are representative of the overall set, increasing the signal-to-noise ration
- identify the significant part(s) of a sequence that triggers a specific error
- define the minimal sequence to trigger a specific error
- possibly: incorporate user feedback in the clustering cycle to iteratively improve results (e.g. semi-supervised learning).

Prototype and evaluate these techniques on real life data from for example advanced railway or banking applications. Determine how to improve existing mining and clustering techniques for test analysis. Indentify future research.

Assignment contact: Menno Jonkers

5. Post-processing of model based test results

Axini develops, applies and sells model based testing tools. Model based testing (MBT) is a new and advanced approach to fully automatically test systems. Based on models (formal descriptions of system behavior) our tools simultaneously generate and execute tests. One of the advantages over existing test approaches is that MBT can do large volume tests: a lot more tests in same amount of time. This gives more insight in the quality of the system under test.

A large number of tests results in a large amount of test data. We need more tools and methods to analyze these results. This assignment addresses the analysis of test results that are obtained from MBT. The major goal is to obtain (partial) answers to some of these questions:

- How should we systematically interpret model based test results, preferable based on a mathematical foundation?
- Can we further automate analysis of test results?
- How can we use the test result to diagnose the erroneous system?
- Can we reduce the test results with aditional knowledge of the model?
- How can we relate test data to customer system requirements?

We expect as deliverables a literature study about the state of the art, and some practical application in Axini's domain with a proof of concept, preferable a case study or prototype tool.

Assignment contact: Rene de Vries

6. STS model checking

Axini develops, applies and sells model based testing tools. Model-based testing (MBT) is a new and advanced approach to fully automatically test systems. Based on models (formal descriptions of system behavior) our tools simultaneously generate and execute tests.

Model-based testing involves making a (formal) model. We make these models manually. Because most of the systems we model are complex, the models tend to become big and complex as well. This makes it hard for people to understand them and to verify whether they are correct.

There is an entire research area called model-checking that is about properties and correctness of models. However these techniques are not applicable out of the box for model-based testing since our models handle a lot of data. To cope with this, we use Symbolic Transition Systems (STS). There is no complete model-checker that can check STSs. Previous work has produced a model checker that can handle a subset of all STS models, but it cannot handle most of the models that we work with.

This assignment is about extending our model-checker so that it can handle all, or at least most, of the models that we write. This is new work, because there is no existing checker that can handle all symbolic models, and it is interesting because of the famous state-space explosion problem.

Possible research questions:

- how do we model check symbolic models that are infinitely large?
- how do we model check symbolic models that use data?
- are there properties for which it is better to test them (with our automated test tools) rather than formally verify them? Which properties are interesting in the context of model based testing?
- how can we simplify our models to make checking them more tractable, while preserving the relevant properties?

Expected deliverable: proof of concept of a (partial) solution that can be used to extend the existing prototype.

Assignment contact: Kevin de Berk

7. Visualization of transition systems

Axini is a company that develops Model Based Testing tools and applies these in projects of our costumer. Model Based Testing (MBT) is a new and advanced approach to test systems. Based on models (formal description of systems) it generates and execute test simultaneously.

The models used by our tools are described by so called Symbolic Transitions Systems (STS). These transitions systems can be analyzed by means of exploration, simulation and visualization.

Axini uses visualizations to give users a view of the models they have created. Unfortunately the benefit of a visualization is reduced when it becomes large and contain lots of information, such as multiple types of labels per transition.

There are abstractions that make it possible to provide higher level views of a model. Information can be hidden, to be revealed when 'zooming in'. The question: what are possible abstractions for this problem and how can they be presented to be most useful?

Assignment

Axini already uses some abstractions in their models and visualizations, such as the ability to group sets of transitions. Your challenge is to come up with new insights into how the information in models can be partitioned and demonstrate how this can be used to present the information in more comprehensible ways, possibly in combination with other analysis tools.

We expect as deliverables an extended visualization method and some practical application in Axini's domain with a proof of concept, preferable a case study or prototype tool.

Assignment contact: Ivo Wever

8. Visualization-based modeling versus textual models

Problem

Axini currently has a textual model editor, because textual models are usually very easy to refactor and to share with others. However, they can be hard to understand for someone who isn't a programmer. Visual models are usually easier to understand, but usually harder to maintain.

To be researched

How can we have best of both worlds?

Determine how existing visualization-based modeling techniques, such as UML or POOSL can be used or extended for using them for model-based testing.

Related work

Example: "POOSL: An Object-Oriented Specification Language for the Analysis and Design of Hardware / Software Systems" - J.P.M Voeten, Research Report 1995

Possible keyworks for literature study: Poosl, UML, UPPAAL, visual modeling

Evaluation

One of our clients is the leading manufacturer in self-scan checkouts. You will apply the developed technique to the systems of this client. You will compare the models and generated tests of the developed technique with the results from non-visual modeling.

Context

The challenge is to build models that on the one hand provide a (high level) system overview, and on the other hand capture all details needed for thorough testing and test derivation.

9. Modeling and testing with decision tables

Problem

Formal modeling of functions in a pragmatic and user friendly way is difficult in a transition system setting. Decision tables are an interesting formalism that seem to enable user friendly (as in also for non-technical people) modeling of functions in a formal way. They are Turing complete and in that sense should be even powerful enough to model entire systems. We are very interested in how we can apply decision tables for model-based testing (and programming in general).

Related Work

- Mors
- Lew, King
- CODASYL82, a modern appraisal of decision tables
- Frantzen, symbolic transition systems

To be researched

How can we use decision tables as a formalism for model-based testing? How do we derive test-cases for decision tables? Is system modeling with decision tables a viable approach or does it become too complex? Can we combine our existing transition system approach with decision tables, getting the benefit of both worlds?

Evaluation

After the literature study you will build one or more prototypes to test your approach. We have several transition system models to compare your approach against. As far as we know this topic has not been researched before, which makes it possible to make a scientific paper out of the results.

10. Model Learning

Problem

In order to test if a system is working correctly, we need a model of its behavior. When a system is already implemented, we could use a method that can automatically obtain such a model. Unfortunately, no such method exists yet for real life situations.

To be researched

Develop a learning technique to learn a model that can be used for model-based testing by observing its behavior and evaluate it. Challenges are non-deterministic systems, systems with concurrent behavior and systems that deal with complex data.

Evaluation

The learning method will be used to obtain models for systems of clients of Axini in order to lower the threshold of starting with model-based testing.

Context

Axini uses Symbolic Transition Systems for modeling. Most existing learning techniques use other forms of modeling and do not support the complexity of practical usage. This project can be done in collaboration with the ITALIA project of the University of Nijmegen. ITALIA is a big research program where other PhD students and Master students are working on model-learning puzzles http://www.italia.cs.ru.nl/)