ONTOLOGY-BASED SOFTWARE TEST CASE GENERATION (OSTAG)

STEW Workshop, 13th October 2016

Vladimir Tarasov, Jönköping University

PROJECT

- **Funded by the Knowledge Foundation**
  - 2015-2017
- **Industrial partners**
  - Saab Avionics (Jönköping)
  - AddQ (Gothenburg)
  - Knowit (Jönköping)
- **Project team from Jönköping University**
  - Vladimir, He Tan, Anders Adlemo, Anders Andersson, Muhammad Ismail
- **http://ju.se/en/research/research-groups/computer-science-and-informatics/research-projects/ontology-based-software-test-case-generation-ostag.html**

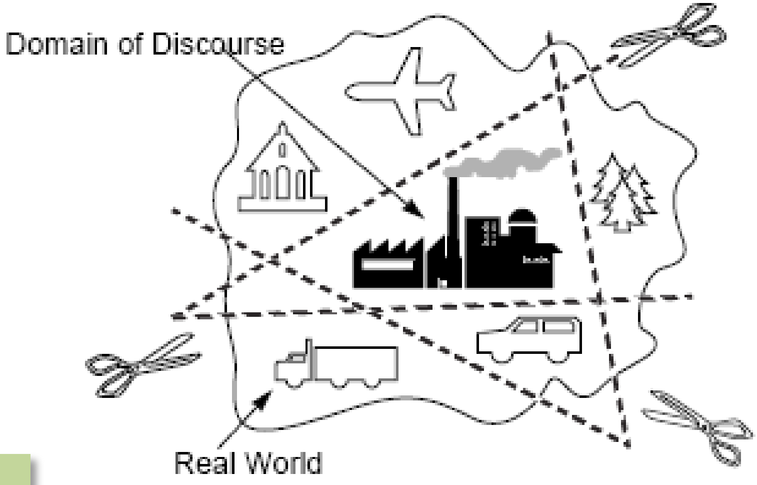
OBJECTIVES

- **Research objective**
  - Create a method for deriving test case data (semi-)automatically using an ontology representing the specification and domain for a software system
- **Technical objective**
  - Develop a prototype of the tool that implements the method and experiment with it
- **Business objective**
  - Make the testing process more effective in terms of resources, time, money, test coverage, as well as in terms of providing additional help to inexperienced testers

APPROACH

- Software Requirement Specification
- Other documents
- Expertise (testers/developers)

Input data



Ontology

- Semantic models of application domain
- Ontologies and Inference rules
- Black-box testing

Test cases

SRS

Evolution

Testing

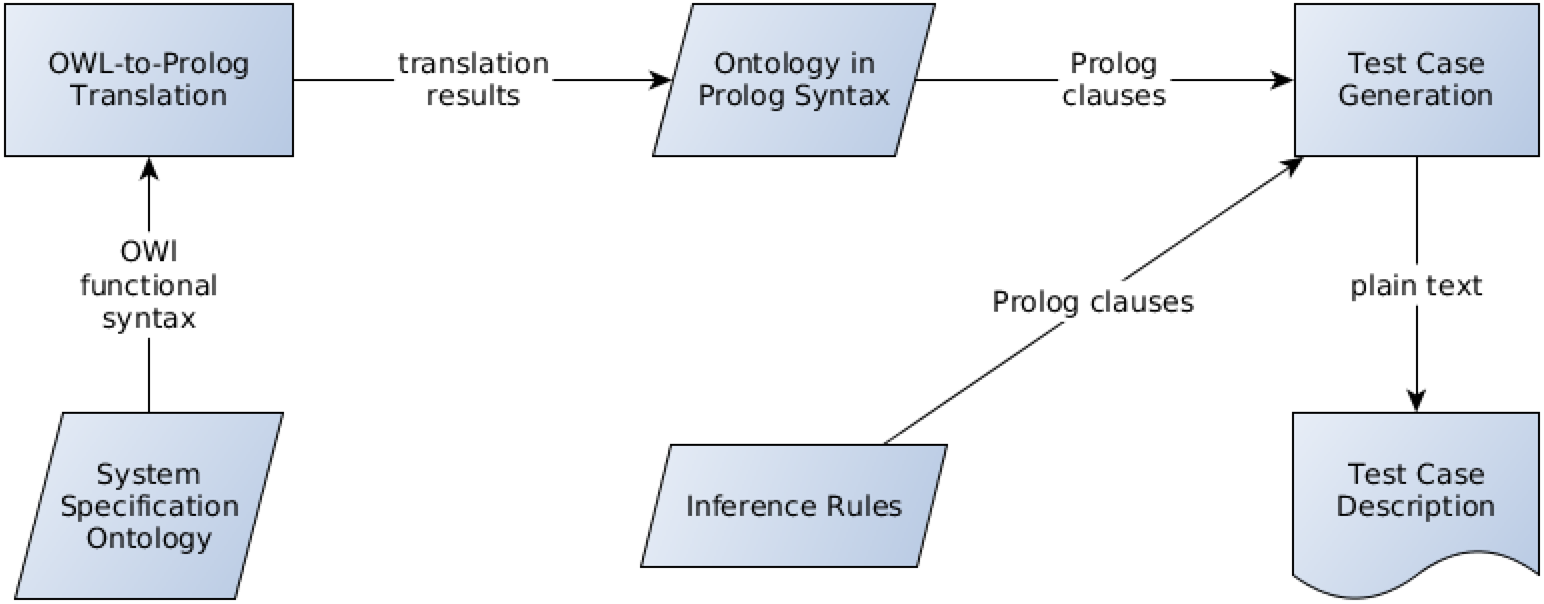
- Evolution algorithms
- Fitness Function
- White-box testing

- Automated generation
- Reduced number of test cases
- Improved test results

CURRENT RESULTS

- **An ontology is developed for the SRS in the SAAB case**
  - Evaluated through SUS and the use in an application
  - Refined in several iterations
- **Inference rules are created to generate test cases**
  - Use the SAAB ontology
  - Almost one-to-one correspondence to the existing test cases
- **Two publications**
  - He Tan, Muhammad Ismail, Vladimir Tarasov, Anders Adlemo and Mats Johansson. *Development and Evaluation of a Software Requirements Ontology*. Accepted to 7th International Workshop on Software Knowledge - SKY 2016.
  - Vladimir Tarasov, He Tan, Muhammad Ismail, Anders Adlemo and Mats Johansson. *Application of Inference Rules to a Software Requirements Ontology to Generate Software Test Cases*. Submitted to OWLED - ORE 2016 - 13th OWL Experiences and Directions Workshop and 5th OWL Reasoner Evaluation Workshop.

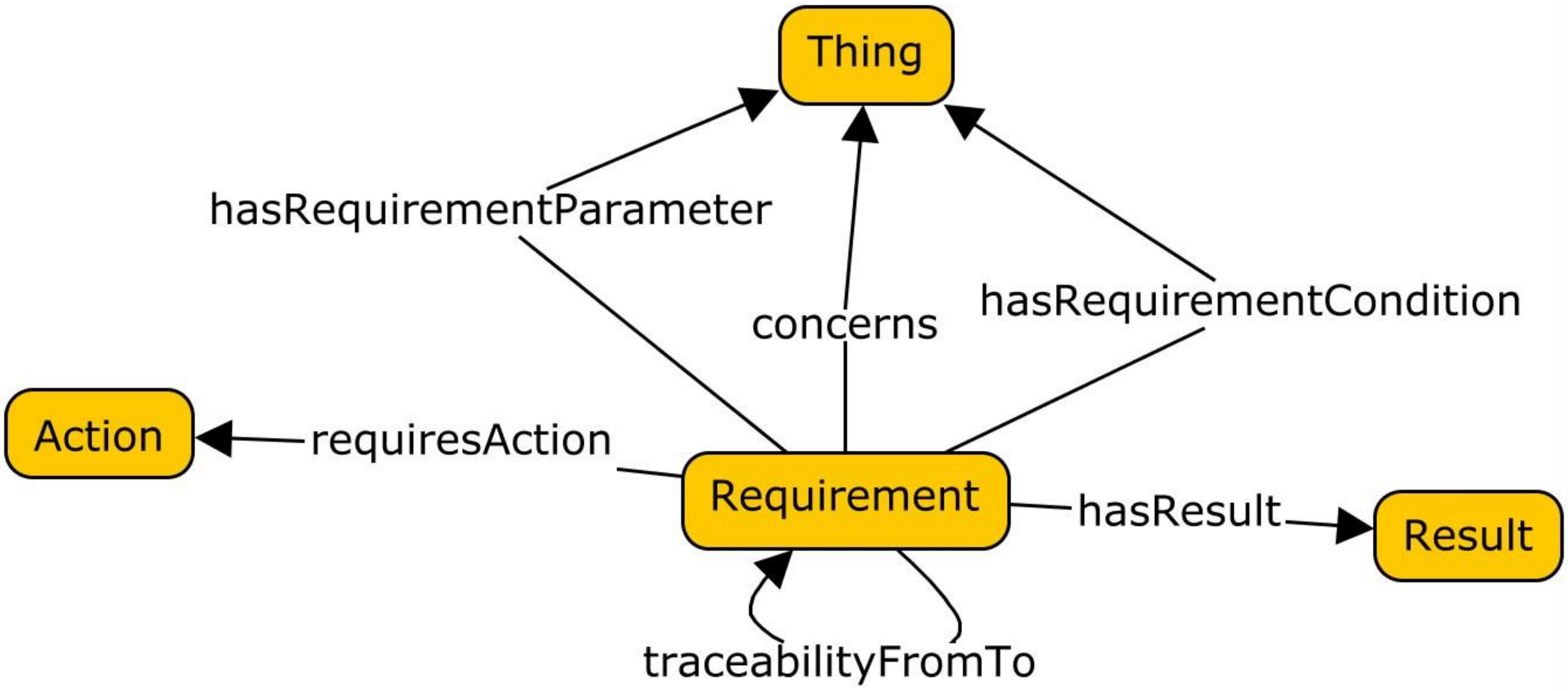
PROCESS OF TEST CASE GENERATION



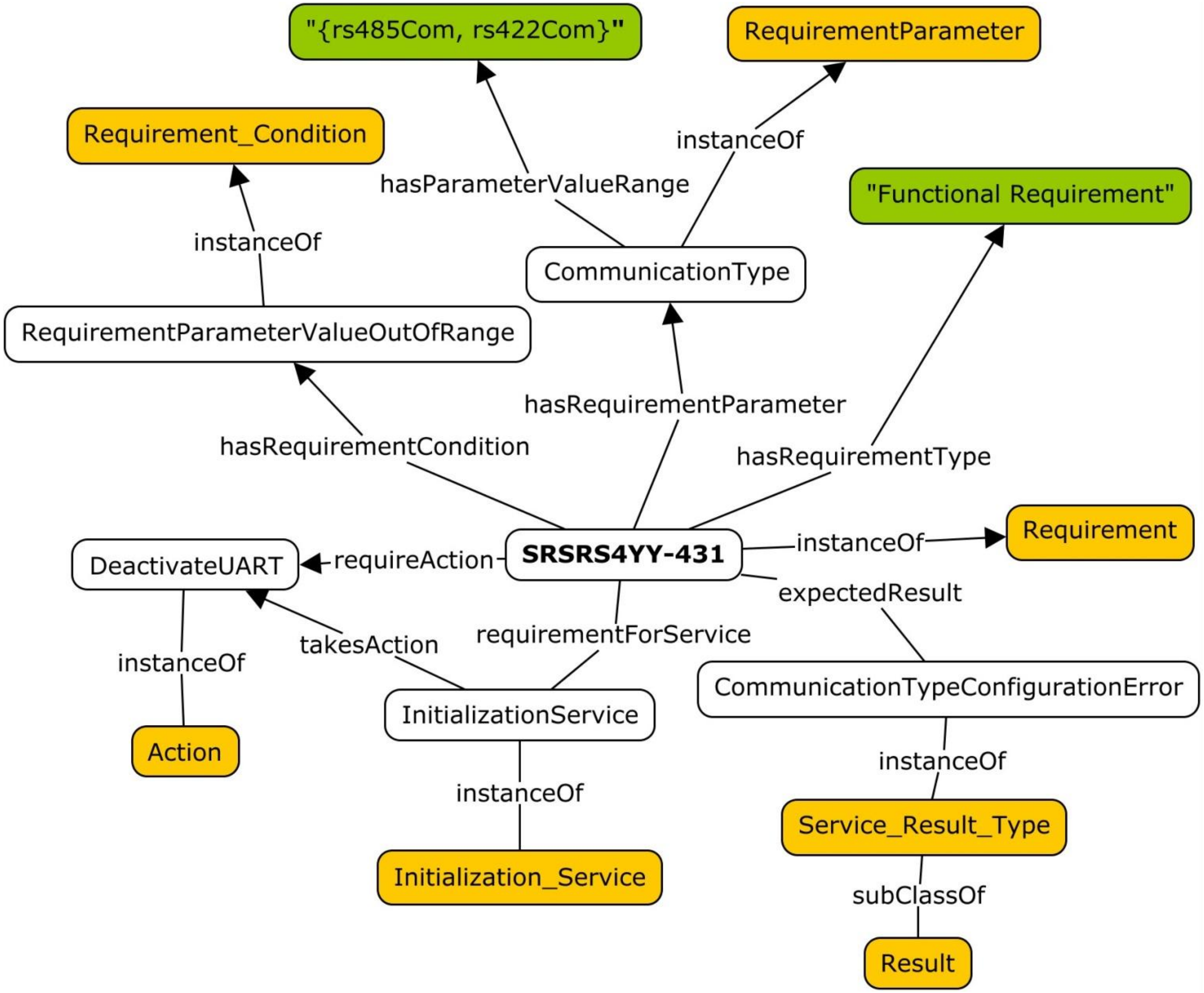
REQUIREMENTS SPECIFICATION ONTOLOGY

- **The ontology includes**
  - A meta model of the software requirements
  - The domain knowledge of the application
  - Each system requirements specifications
- **The ontology contains**
  - 42 classes
  - 34 object properties
  - 13 datatype properties
  - 147 instances in total

THE META MODEL OF A REQUIREMENT



ONTOLOGY FRAGMENT: SRSRS4YY-431

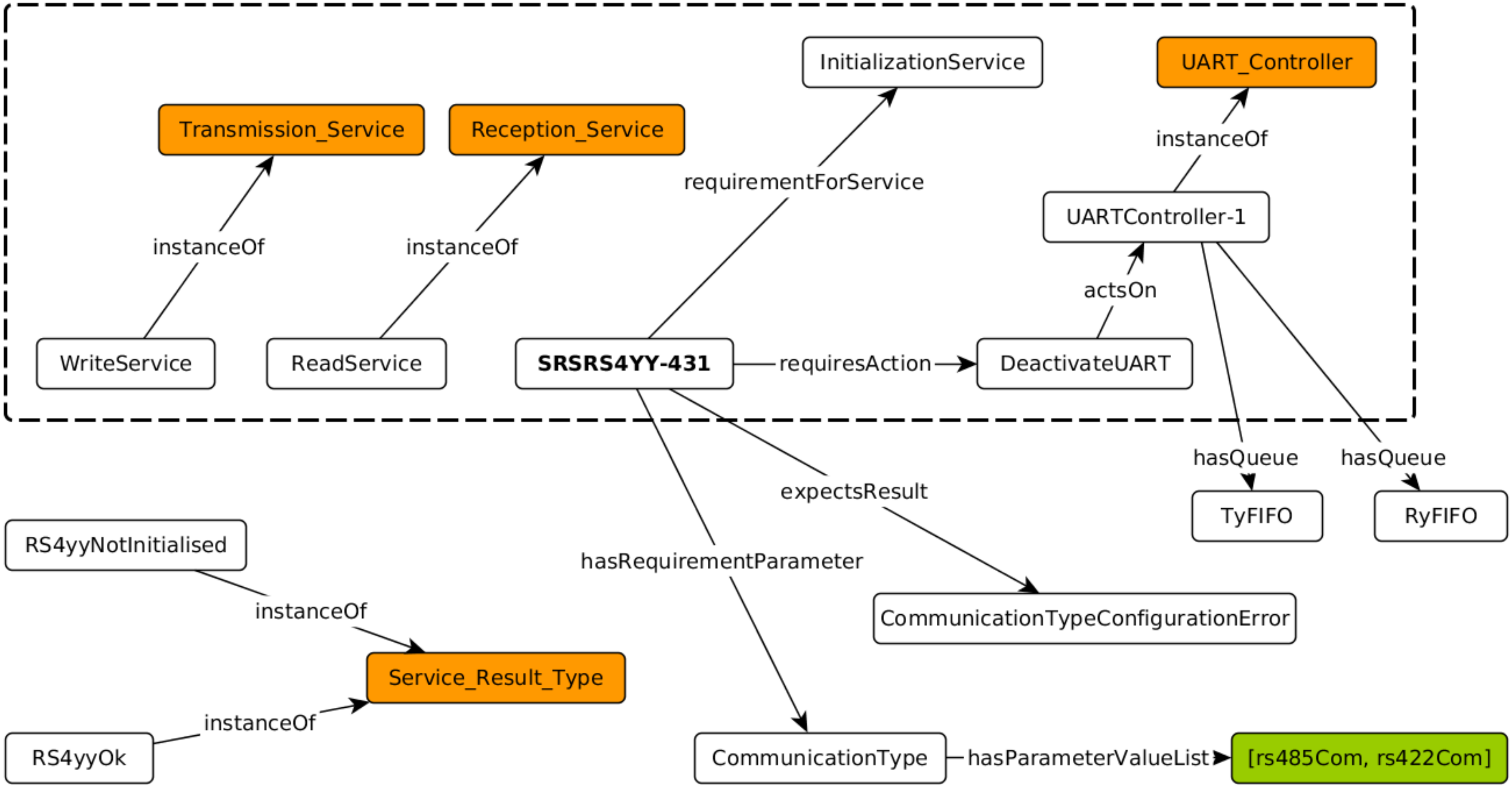


AN EXAMPLE OF A HEURISTIC RULE: SRSRS4YY-431

IF the requirement is for a service and a UART controller is to be deactivated
THEN add the call to the requirement's service, calls to a transmission service and reception service as well as a recovery call to the first service.

```
tc_procedure(Requirement, Procedure) :-
  % get service instance for call #1
  objectPropertyAssertion(requirementForService, Requirement, Service),
  % check condition for calls #2-4
  objectPropertyAssertion(requiresAction, Requirement, DeactivateUART),
  objectPropertyAssertion(actsOn, DeactivateUART, UartController),
  classAssertion(uart_controller, UartController),
  % get instances of the required services
  classAssertion(transmission_service, WriteService),
  classAssertion(reception_service, ReadService),
  Procedure = [Service, WriteService, ReadService, recovery(Service)].
```

ONTOLOGY PATHS USED BY THE INFERENCE RULES TO GENERATE A TEST CASE



DEMO

EXPERIMENT

- 40 inference rules were used to generate the 18 test cases.
- The corresponding test cases have been reproduced in plain English
- Almost one-to-one correspondence between the texts in the generated test cases and the texts provided by one of our industrial partners, Saab

FUTURE WORK

- Using FrameNet as a general lexicon to modell complex requirements
- Express test case generation strategies in terms of algorithms rather than inference rules
- Computing ontology coverage to check requirements coverage