Métodos Formais em Engenharia de Software

Mestrado Integrado em Engenharia Informática e Computação

2017/18

Practical work on VDM++

Goals

The goal of this practical work is to develop, test and document an executable formal model of a high integrity software system in VDM++ using the Overture tool or the VDMTools. At the end of the work, students should have acquired the ability of formally modeling software systems in VDM++, and of demonstrating the consistency of the model. The work is to be performed by groups of two to three students attending the same class (turma). The groups will be defined and the themes assigned (different themes for different groups in the same class) in the practical classes of the week of 27 November to 1 December 2017.

Deliverables

The project should be organized in a way similar to the Vending Machine example (see the MFES page in Moodle). The practical work should be concluded and the deliverables (final report, VDM++ project files, UML project files) submitted in a .zip file through Moodle until **11pm January 3**. It may be applied a penalty of 2 (points out of 20) for each day late. The presentations will be scheduled for the first week of January (4th or 5th of January). Presentation is mandatory for **ALL** students. **If you miss presentation you will get zero points out of twenty.**

Report Structure

You should incorporate the developed VDM++ classes in a single report (to be submitted in PDF format), written in English or Portuguese, covering the following items (which are also the assessment items):

1. Front page with authors, date and context (FEUP, MIEIC, MFES) [1%]
2. Informal system description and list of requirements [10%]
    a. Requirements should include any relevant constraints (regarding safety, etc.).
    b. Each requirement should have an identifier.
    c. You may have optional requirements.
3. Visual UML model [5%]
    a. A use case model, describing the system actors and use cases, with a short description of each major use case.
    b. One or more class diagram(s), describing the structure of the VDM++ model, with a short description of each class, plus any other relevant explanations.
4. Formal VDM++ model [50%]
    a. VDM++ classes, properly commented.
    b. Needed data types (e.g., String, Date, etc.) should be modeled with types, values and functions.
    c. Domain entities should be modeled with classes, instance variables and operations. You are expected to make adequate usage of the VDM++ types (sets, sequences, maps, etc.) and create a model at a high level of abstraction.

    d. The model should contain adequate contracts, i.e., invariants, preconditions, and postconditions. Postconditions need only be defined in cases where they are significantly different from the operation or function body (e.g., the postcondition of a sqrt(x) operation, which simply states that x = RESULT * RESULT, should be significantly different than the body); for learning purposes, you should define postconditions for at least two operations.
    e. During the development of the project, if you foresee that the size of the VDM++ model will be less than 5 pages (or 7.5 pages in case of groups of 3 students) or more than 10 pages (or 15 pages in case of groups of 3 students), you should contact your teacher to possibly adjust the scope of the system or the modeling approach being followed.
5. Model validation (i.e., testing) [20%]
    a. VDM++ test classes, containing adequate and thorough test cases defined by means of operations or traces.
    b. Evidences of test results (passed/failed) and test coverage. It is sufficient to present the system classes mentioned in 4 painted with coverage information. Ideally, 100% coverage should be achieved. Optionally, you may include figures of examples exercised in the test cases.
    c. You should include requirements traceability relationship between test cases and requirements. Ideally, 100% requirements coverage should be achieved. It is sufficient to indicate in comments the requirements that are exercised by each test.
6. Model verification (i.e., consistency analysis) [5%]
    a. An example of domain verification, i.e., a proof sketch that a precondition of an operator, function or operation is not violated. You should present the proof obligation generated by the tool and your proof sketch.
    b. An example of invariant verification, i.e., a proof sketch that the body of an operation preserves invariants. You should present the proof obligation generated by the tool and your proof sketch.
7. Code generation [5%]
    a. You should try to generate Java code from the VDM++ model and try to execute or test the generated code. Here you should describe the steps followed and results achieved.
8. Conclusions [3%]
    a. Results achieved.
    b. Things that could be improved.
    c. Division of effort and contributions between team members.
9. References [1%]

FAQs

Why are my post-conditions similar to the function's body?

Au contraire, you should actually be wondering: "why is the body function a repetition of the post-condition"? In most cases such scenario is perfectly natural; for example, if the postcondition of a function is "the return value is the product of two arguments", then probable the function's body will

use the same arithmetic operations to achieve the end result. However, that's not always the case. Take a function that return the square root of a number. The post-condition could be easily expressed as "the result, whatever it is, when squared, is equal to the input". The body would be much more complex, since it needs to actually "compute" the root. The same thing goes for multiplication; one could say that multiplying A by B is summing B times the number A.

So is one allowed to repeat the body in the post-condition?
For sufficiently complex functions, the repetition only occurs because one tend to first build the function body and then the postcondition. The latter would then assume an "algorithmic" expression, instead of a "declarative" assertion. The rule of thumb is to invert the rationale: first one should build the intended post-condition and only then the specific algorithmic implementation. The philosophy is akin to the difference between TFD (Test-First Development) and TLD (Test-Last Development).

So... one should always design the contracts first?
Absolutely! And before the contracts, one should make the types as strong as possible (they are the de facto first line of formalization). Sufficiently strong types can easily replace a number of invariants, pre and postconditions.

What makes a good pre/postcondition?
Pre-conditions restrict the scenarios where an operation can take place. Hence, a good pre-condition will be one that is as weak as possible to allow the correct functioning of the operation in the widest possible situations, but not weaker to the point where it accepts invalid status. This is commonly known as the weakest precondition. Example: given a function *sqrt: Double → Double*, a possible precondition would be *input>1*. Although valid, it's not the weakest one, since it's discarding 0 and 1 as possible (and perfectly acceptable) inputs. An invalid (though weaker) pre-condition would accept negative numbers as an input, something whose square root cannot be expressed in the domain of Doubles.
Post-conditions specify the ending scenario after an operation took place. Hence, a good post-condition will be one that is as strong as possible to exactly define what the operation will end up doing in the widest possible situations, but not stronger to the point where perfectly valid results would not be accepted. This is commonly known as the strongest postcondition. Example: given a function *primesUntil*: *Nat → Seq of Nat*, a possible post-condition would be that the result should be a subset of all odd numbers up until N (*RESULT in { x | x: nat & x < N and x mod 2 = 1}*). Although all prime numbers are odd, some odd numbers are not prime. Hence, the post-condition should be made stronger.

Themes

1- Web Summit

In this project you have to model all the information needed to manage the Web Summit event. There are different conferences, attendees, speakers, schedule, startups and news. It should be possible to determine the events available in a specific hour, the number of attendees, among other things.
See https://websummit.com

2- Safety Net Hospital

In this project you have to model all the information needed to manage a Safety Net Hospital. **Safety net hospitals** provide care for populations which are of low-income, traditionally exploited, receiving public assistance, and uninsured. Safety net hospitals are not defined by their ownership terms; rather they are more devoted to providing the best possible care for those who are barred from health care due to various circumstances. With this system, it should be possible to know the doctors of an hospital, the doctors that work in more than one hospital, the hospital with higher number of appointments, among others.
See https://en.wikipedia.org/wiki/Safety_net_hospital

3- Rally Competitions

In this project you should model all information needed to manage different rally competitions. Rally is the leading, GPS enabled, competition grade application for use in almost all types of rally including time trial, special stage, regulatory, TSD, historic and vintage rallies.
Rally supports single and multi-day rallies containing an almost unlimited number of sections containing all types of time and passage controls, grouping, service, refuel, holding, locality, special and time trial elements as well an unlimited number of in sector average speed, time and distance changes. Competitions are in different places with possibly different participants. A competition may have different steps and, at the end, a winner. It should be possible to know all the time spent by each participant, the winners of different competitions, the sponsors, among others.
See https://play.google.com/store/apps/details?id=com.crml.android.rally

4- Fashion Shows

In this project you should model all the information needed to manage different fashion shows. Different shows happen in different places on different dates. It should be possible to know the fashion designers present in each show; the theme of their show; which are the fashion designers participants in each show; among others. In addition, it is possible to connect users with the latest runway images (more on that in a moment) and help attendees work their way through the myriad of shows, presentations, and events to booking last-minute primping sessions or snapping up new ensembles on the fly—and all without breaking a sweat.
See https://www.vogue.com/article/10-fashion-week-apps

5- Stack Overflow

In this project you should model all the information managed by the Stack Overflow website. This website presents questions and answers on a large number of computer programming topics. The

website serves as a platform for users to ask questions and respond to them. But in addition they can, through registration and active participation, vote on more or less useful questions and answers.
See https://pt.wikipedia.org/wiki/Stack_Overflow

6- Fitness App

In this project you should model all the information managed by a Fitness App. With this app users can search for nearby running routes or share their favorite routes. Map My Fitness saves data on pace, distance, and calories burned for GPS-based workouts; you can use this data to set new personal goals. Through the Map My Fitness community, users can join challenges as well as motivate friends.
See https://www.pcmag.com/article2/0,2817,2485287,00.asp

7- Indoor Finder

In this project you should model all the information needed for an app that is able to guide you inside buildings. It should be possible to represent the plant of a building or set of buildings such as FEUP to offer functionalities such as:
- advise the shortest route between two interior points;
- advise the most accessible route between two interior points (for example, wheelchair users use lifts instead of stairs);
- find a type of equipment closer to a given position (e.g., WC, coffee machine, printer).

8- Small Business Discount Network

In this project you should model all the information and transactions managed by a small business discount network. Small merchants can join the system. Consumers / customers can join the system by receiving a discount card (physical or virtual) that they use in purchases from affiliated merchants. In each purchase, there is a charge that is charged to the merchant (for example, 5% of the value of the sale), a part of which is returned to the customer (for example, 3%) and the remainder is a service charge to pay the maintenance of the system (e.g., 2%). The customer may use his balance on subsequent purchases from any affiliated merchant. You can also transfer part of your balance to other adherent customers. Customers who get new customers or merchants who raise new merchants also receive bonuses. Merchants can offer extra discounts on selected products (increasing the % that reverts to the customer).
See: http://www.boobonus.com/

9- Brand and Celebrities

This project aims to model all the information managed by Brand and Celebrities app. Brand and Celebrities created the first platform dedicated to Celebrity Marketing, allowing brands and agencies to identify, analyse and contact the celebrities that are the best fit for their projects (brand ambassador, speaker, entertainer, event sponsor, digital influence, product placement…), but also to finalise a contract with them easily.
See https://brandandcelebrities.com/en/our-vision/

10- Ugly Fruit

This project aims to model all the information managed by Ugly Fruit. About half of the food produced in the world each year goes to waste. The reasons for this waste are many and occur throughout the agrifood chain. The “Fruta Feia” cooperative arises from the need to reverse such fruit

and vegetable standardization trends that have nothing to do with food safety and quality issues. This project aims to combat market inefficiency by creating an alternative market for "ugly" fruit and vegetables that can change consumption patterns.
See: http://frutafeia.pt/pt/projecto

11- MagicStay

This project aims to model all the information managed by MagicStay. MagicStay is the first alternative accommodation booking site, entirely dedicated to business travelers. MagicStay offers accommodation selected to meet the needs of business travelers, ideally located in city centers or near the congress / exhibition venues. MagicStay allows to save money, have more surface and privacy than a hotel, including a kitchen, but also to travel with colleagues or extend your stay with family.
See https://www.magicstay.com/

12- BUYINPORTUGAL.PT

This project aims to model all the information managed by BUYINPOTUGAL.PT. BuyinPortugal.pt is a business-to-business e-commerce website, that facilitates the sale of Portuguese sellers, from small and medium enterprises, to buyers around the world, and operates in English. Its mission is to make it easy to do business with Portugal, by helping business buyers find products and services from Portuguese suppliers, quickly and efficiently.
See https://buyinportugal.pt/index.php

13- Hokify

This project aims to model all the information managed by Hokify. Hokify provides you with a large selection of jobs that match your interests and skills. You can create your CV and look for jobs.
See https://hokify.de/jobs

14- Iron Trainers

This project aims to model all the information managed by Iron Trainers.
Training, nutrition and health in your hands. A system developed by a team that involves physicians, personal trainers, psychologists and many other professionals, Iron Trainers brings to the user an innovative and unique experience around the world. Through a mobile application with predefined interfaces, Iron Trainers allows the user to collect important data about their health and sports practice. From nutrition control to a complete and intelligent training system, Iron Trainers makes the user's smartphone a powerful tool for planning and delivering results.
See https://irontrainers.pt/about-us/

15- Jyve

Jyve empowers local musicians and bands by connecting them to venues through a mobile app. Through the app you'll be able to create a profile, connect with musicians, sync your calendar, find gigs and manage them all in one place. At Jyve we strive to solve the common problems in the local music scene by creating the one stop app for musicians, bands, and venues.
See https://jyve.io/