Simon Jozsef-Gabriel
IS1.1B

Set de Metrici

1. Introducere

Pentru a calcula setul de metrici pe care urmeaza sa il prezint, am folosit urmatoarele tool-uri:

- **SonarQube:** o platforma open-source implementata de SonarSource, folosita pentru inspectia continua (en. Continuous inspection) a calitatii codului prin executarea unor revizuiri automate folosind analizarea codului pentru a detecta bug-uri, code smells, si vurnerabilitati de securitate pentru mai multe de 20 de limbaje de programare. SonarQube ofera rapoarte asupra codului duplicat, standardelor codului, testelor unitare, complexitatea codului, comentarii, bug-uri si vulnerabilitati de securitate.
- **SDMetrics:** este un tool pentru UML care masoara calitatea design-ului. Practic, SDMetrics analizeaza structura modelelor UML. Printre metricile orientatea pe obiect se numara, in principal, marimea design-ului, cuplarea codului si complexitatea.

2. Setul de Metrici

In continuare am sa prezint cele 17 metrici calculate cu ajutorul tool-urilor mentionate mai sus, asupra aplicatiei dezvoltate de mine, si anume: evidenta cartilor dintr-o biblioteca. Pentru cod am folosit Java, deci metricile sunt calculate in conformitate cu acest limbaj, si Maven pentru dependinte.

- **NumOps** – numarul de operatii sau metode dintr-o clasa. In total, am implementat 190 de metode in cadrul celor 40 de clase, deci o medie de 4.75 de metode per clasa. Numarul recomandat de metode per clasa este de mai putin de 20. In cazul meu, am doar o singura clasa cu mai multe de 20 de metode, ceea ce va putea fi fixat cu usurinta prin divizarea clasei.
- **DIT** – este adancimea arborelui de mostenire. Ca si valori, am obtinut un total de 14 si o medie de 0.42 per clasa. Valoarea recomandata este sub 6.

- **NumAttr** – numarul de atribute/campuri/variabile. Am obtinut un total de 114 atribute, cu o medie de 2.85 per clasa. Valoarea recomandata este de sub 6 per clasa.
- **Code Smells** – simptome ale codului sursa al programului care pot indica o problema mai profunda. In cazul meu, am 10 code smells, iar presupun ca valoarea recomandata pentru aceasta metrica ar fi 0 sau cat mai mica.
- **Linii de cod** – reprezinta numarul total de linii de cod ale programului. Aplicatia mea are 1926 de linii de cod, deci o medie de 49.38 de linii de cod per clasa. Valoarea recomandata ar fii de sub 1000 de linii de cod per clasa.
- **Statements** – reprezinta numarul de declaratii. Aplicatia a obtinut un numar total de 739 de declaratii. Din pacate, nu am putut gasi o valoare recomandata pentru aceasta metrica.
- **Clase** – numarul total de clase din cadrul codului sursa al aplicatiei. Am obtinut un numar total de 40 de clase. Nu am reusit sa gasesc o valoare recomandata, insa am vazut aplicatii cu zeci de mii de clase, deci presupun ca nu exista.
- **Comentarii** – numarul de linii ce reprezinta comentarii din cadrul codului sursa. Am obtinut un numar de 124 de linii, adica 6% din toate liniile de cod. Aceasta valoare ar trebui sa fie de cel putin 30%, prin urmare, va trebui sa adaug cat mai multe linii de comentarii pentru ca si codul meu sa fie cat mai bine inteles.
- **Cyclomatic Complexity**- aceasta complexitate este calculata dupa numarul de cai din cod. Ori de cate ori fluxul de control al unei functii este impartit, aceasta complexitate creste cu 1. In cazul meu, se iau in considerare declaratiile if, while, case, catch, throw, && etc. Aplicatia a obtinut valoare de 313. Aceasta valoare, presupun, ar trebui sa fie cat mai mica, deci va trebui sa fac unele ajustari la cod pentru a o rectifica.
- **Cognitive Complexity** - reprezinta cat de greu este sa intelegi fluxul de control al codului. Am obtinut o valoare de 109, insa cred ca aceasta va trebui sa fie mai mica dupa ce o sa aplic unele modificari.
- **Technical Debt** – necesarul de efort pe care trebuie sa il depun pentru a fixa code smell-urile. Este masurata in minute. Am obtinut o valoare de 1h 12min, ceea ce cred eu ca e rezonabil, tinand cont de faptul ca

nu ar trebui nici macar o zi de munca pentru a rezolva aceste probleme.

- **Technical Debt ratio** – proportia dintre costul de dezvoltare al aplicatiei si costul de fixare. Din moment ce eu nu am atat de multe probleme, aceasta proportie este de 0.1%.
- **R** – numarul de relatii dintre clase si interfete din cadrul unui pachet. Am obtinut valoare de 39 de relatii, adica o medie de 2.29 per pachet.
- **H** – coeziunea relationala. Reprezinta numarul mediu de relatii interne per clasa/interfata si este calculat ca raportul dintre R+1 si numarul de clase si interfete din pachet. Prin urmare, am obtinut valoarea de 22.51 de relatii, cu o medie de 1.32 per pachet.
- **Ca** – cuplajul aferent. Numărul de elemente din afara acestui pachet care depind de clase sau interfețe din acest pachet. Dependențele considerate sunt aceleași listate cu metrica R. Aplicatia a obtinut valoarea de 24 in total, si 1.41 per pachet, ca si medie.
- **Ce** – cuplaj eferent. Numărul de elemente din afara acestui pachet de care depind clasele sau interfețele din acest pachet. Dependențele considerate sunt aceleași listate cu metrica R. Aplicatia a obtinut in total 86 de elemente, cu o medie de 5.05 per pachet.
- **I** - Instabilitate sau ușurință în schimbare. Acesta este raportul dintre cuplajul eferent (metric Ce) și cuplajul total (Ce+Ce). Valorile metricii I variază între 0 și 1. O valoare apropiată de 0 indică un pachet care nu se bazează prea mult pe alte pachete, dar alte pachete se bazează foarte mult pe acest pachet. Un astfel de pachet ar trebui să fie stabil, deoarece este greu de schimbat: modificările aduse pachetului pot avea un impact mare asupra modelului. O valoare apropiată de 1 indică un pachet care se bazează în mare parte pe alte pachete, dar pe care în sine nu se bazează prea mult. Un astfel de pachet poate fi instabil, deoarece este ușor de schimbat: modificările aduse pachetului nu sunt de natură să aibă un impact mare asupra modelului. Ca si valori, aplicatia mea a obtinut media de 0.85, ceea ce inseamna ca se apropie mai multe de cel de al doilea caz, adica majoritatea pachetelor sunt instabile.

3. Concluzie

Ca si concluzie, pot spune ca aplicatia mea are unele probleme, insa ele sunt destul de usor de fixat, si nu cred ca imi va lua mai mult de 8h pentru a reusi sa aduc majoritatea metricilor prezentati la valorile recomandata sau cat mai normale. In cazul unor metrici, nu am reusit sa gasesc valorile recomandate, insa, vom vedea cum vor iesi rezultatele si dupa ce voi modifica codul sursa.

4. Referinte

- https://docs.sonarqube.org/latest/user-guide/metric-definitions/
- https://en.wikipedia.org/wiki/SonarQube
- https://www.sdmetrics.com/manual/Metrics_class.html
- https://www.sdmetrics.com/LoM.html
- Software Metrics for Design and Complexity din cadrul fisierului de pe Classroom
- https://ro.wikipedia.org/wiki/Code_smell