

Tim 6 - Softverske Metrike



Članovi tima

Članovi:

- Petar Turanović 74/19
- Marko Popović 600/21
- Ružica Gagić 665/21
- Vukić Leković 84/19

~~— Željana Pujin 71/19~~

*Napomena, detaljnije o ovome [ovde](#)

Šef tima:

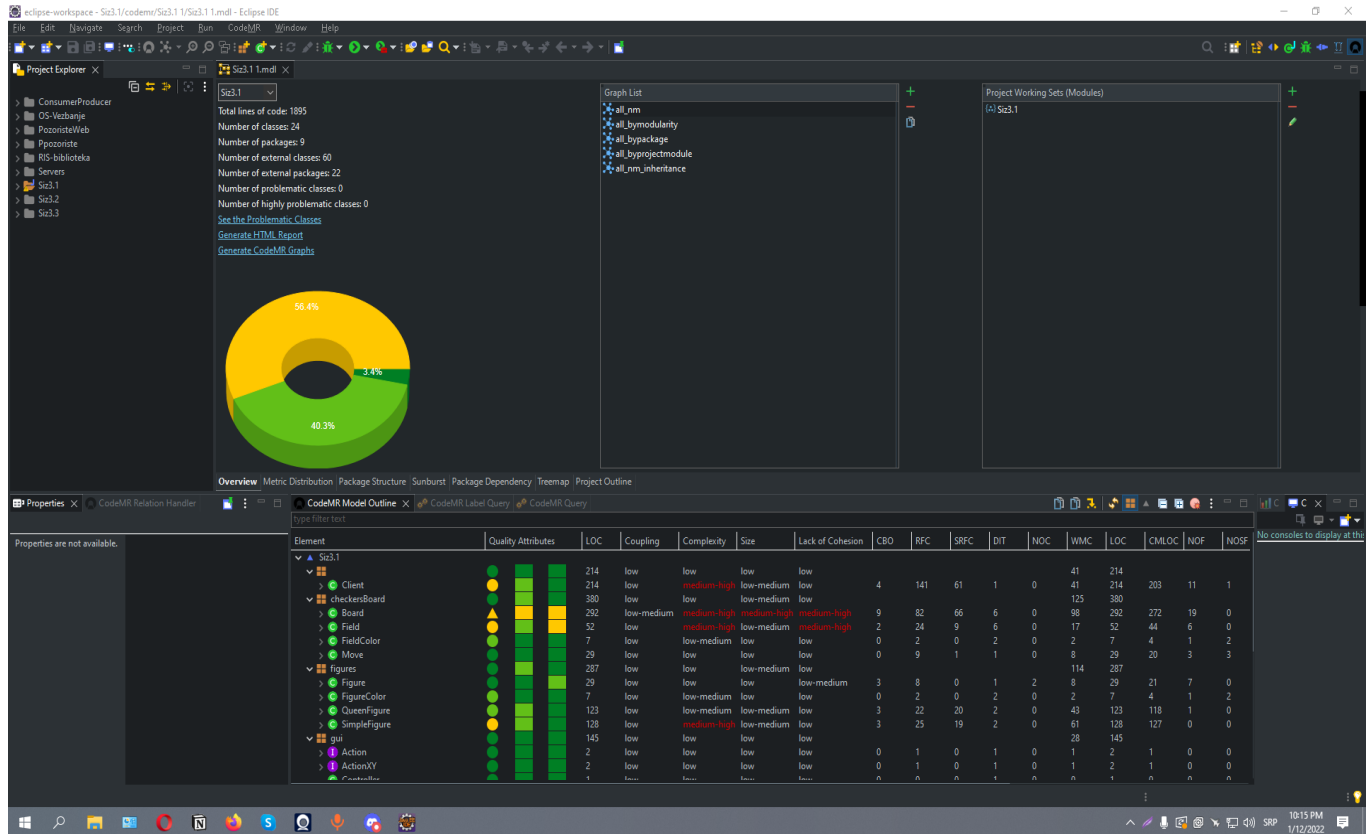
- Uroš Đerić 46/19

Opis korišćenih alatki

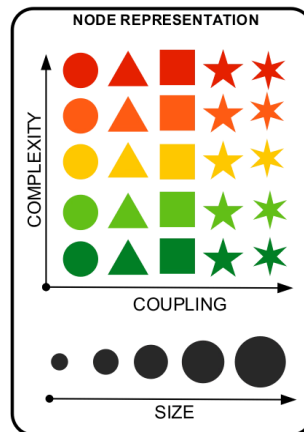
- [CodeMR](#)
- [Metrics2](#)
- ~~[AssessStyle](#)~~
- [CheckStyle](#)
- [Statistic](#)

1. CodeMR

- Predstavlja glavnu alatku koju smo koristili u merenju većine softverskih metrika u datom zadatku
- Alatka poseduje GUI prikaz parametara metrika u formi pie chart-ova kao i u formi tabelarnog prikaza



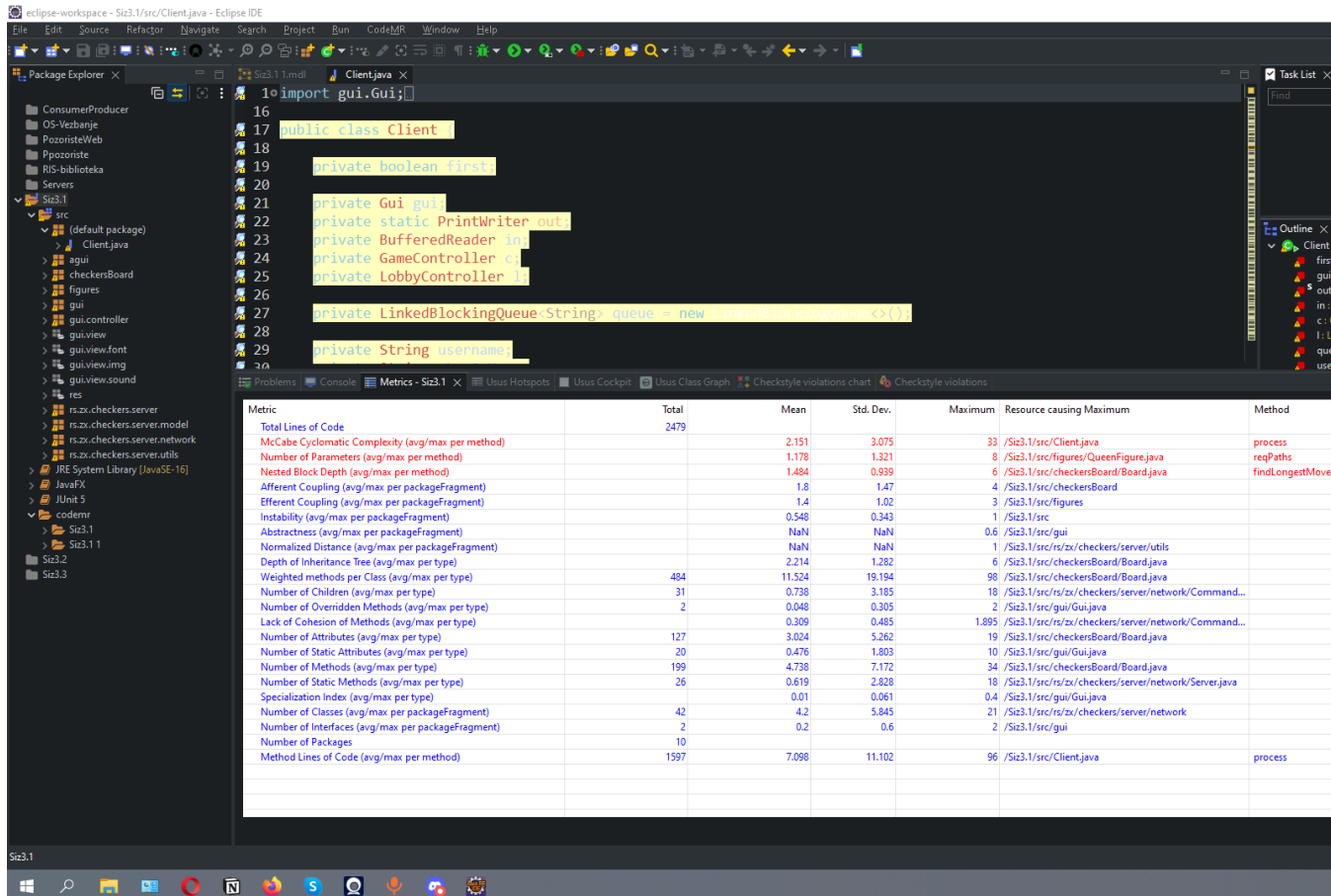
- Slika 1.1. Prikaz rada Code MR plugin-a



-Slika 1.2. Prikaz legende grafičkog trenda kompleksnosti

2. Metrics2

- Pomoćni Eclipse plugin kojim smo se služili da odredimo neke metrike koje [Code MR](#) nije pokazivao
- Alatka poseduje detaljni tabelarni prikaz parametara metrika



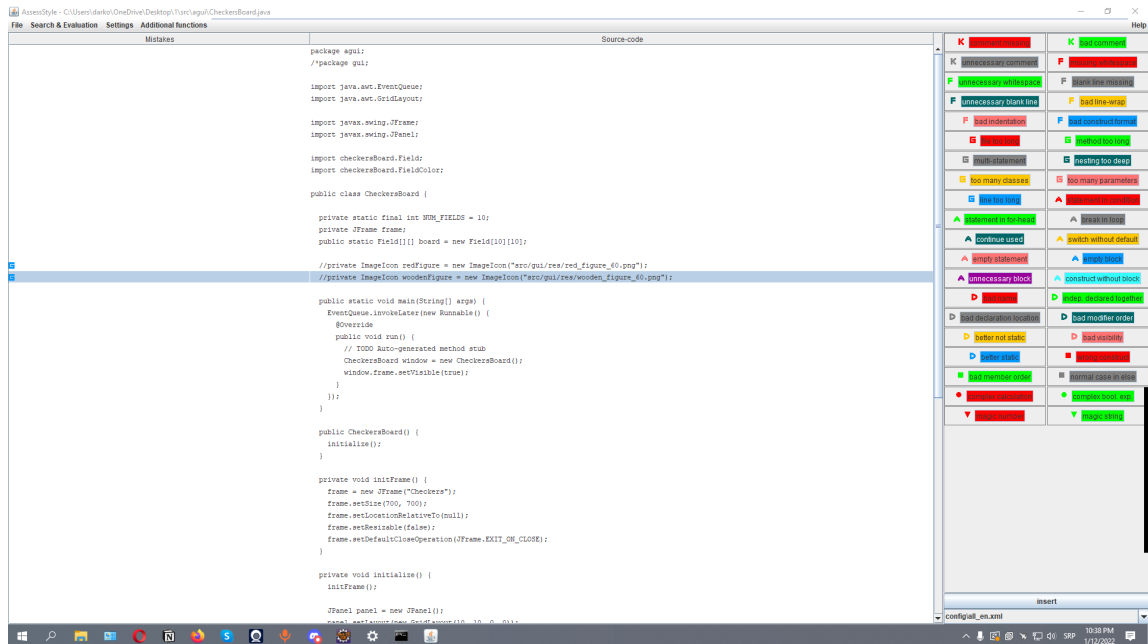
The screenshot shows the Eclipse IDE with the Metrics2 plugin. The main editor displays the source code of `Client.java`. The bottom panel shows a table of metrics for the project.

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Total Lines of Code	2479					
McCabe Cyclomatic Complexity (avg/max per method)		2.151	3.075	33	/Siz3.1/src/Client.java	process
Number of Parameters (avg/max per method)		1.178	1.321	8	/Siz3.1/src/figures/QueenFigure.java	reqPaths
Nested Block Depth (avg/max per method)		1.484	0.939	6	/Siz3.1/src/checkersBoard/Board.java	findLongestMove
Afferent Coupling (avg/max per packageFragment)		1.8	1.47	4	/Siz3.1/src/checkersBoard	
Efferent Coupling (avg/max per packageFragment)		1.4	1.02	3	/Siz3.1/src/figures	
Instability (avg/max per packageFragment)		0.548	0.343	1	/Siz3.1/src	
Abstractness (avg/max per packageFragment)		NaN	NaN	0.6	/Siz3.1/src/gui	
Normalized Distance (avg/max per packageFragment)		NaN	NaN	1	/Siz3.1/src/rs/zs/checkers/server/utis	
Depth of Inheritance Tree (avg/max per type)		2.214	1.282	6	/Siz3.1/src/checkersBoard/Board.java	
Weighted methods per Class (avg/max per type)	484	11.524	19.194	98	/Siz3.1/src/checkersBoard/Board.java	
Number of Children (avg/max per type)	31	0.738	3.185	18	/Siz3.1/src/rs/zs/checkers/server/network/Command...	
Number of Overridden Methods (avg/max per type)	2	0.048	0.305	2	/Siz3.1/src/gui/Gui.java	
Lack of Cohesion of Methods (avg/max per type)		0.309	0.485	1.895	/Siz3.1/src/rs/zs/checkers/server/network/Command...	
Number of Attributes (avg/max per type)	127	3.024	5.762	19	/Siz3.1/src/checkersBoard/Board.java	
Number of Static Attributes (avg/max per type)	20	0.476	1.803	10	/Siz3.1/src/gui/Gui.java	
Number of Methods (avg/max per type)	199	4.738	7.172	34	/Siz3.1/src/checkersBoard/Board.java	
Number of Static Methods (avg/max per type)	26	0.619	2.828	18	/Siz3.1/src/rs/zs/checkers/server/network/Server.java	
Specialization Index (avg/max per type)		0.01	0.061	0.4	/Siz3.1/src/gui/Gui.java	
Number of Classes (avg/max per packageFragment)	42	4.2	5.845	21	/Siz3.1/src/rs/zs/checkers/server/network	
Number of Interfaces (avg/max per packageFragment)	2	0.2	0.6	2	/Siz3.1/src/gui	
Number of Packages	10					
Method Lines of Code (avg/max per method)	1597	7.098	11.102	96	/Siz3.1/src/Client.java	process

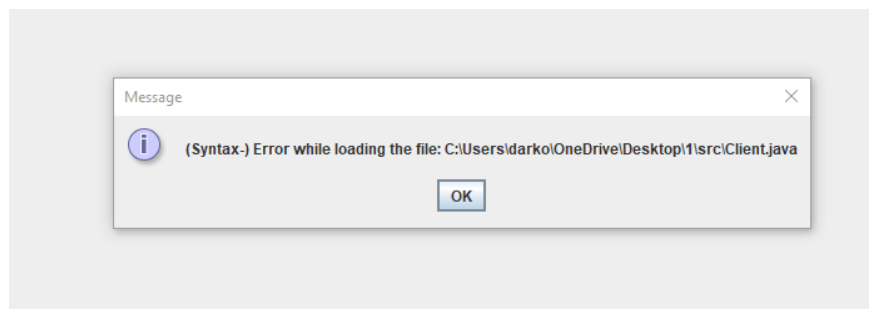
Slika 2.1. Prikaz rada Metrics2 plugin-a

3. AssessStyle

- Third-party softver koji je osmišljen da meri metriku stila na automatski i polu-automatski način
- Dokazao se kao grozan, nepredvidljiv softver na koji smo protraćili sat vremena pokušavajući da ga osposobimo za rad



- Slika 3.1. Prikaz softvera u radu (u retkim situacijama u kojima bi radio)



- Slika 3.2. Prikaz Softvera u (ne)radu (mnogo češća pojava)

- Zbog svoje zastarele i nepredvidljive prirode preporučujemo da ga uklonite sa slajdova na kome se spominju predloženi alati za metrike stila

4. CheckStyle

- Pomoćna alatka kojom smo se služili kako bismo odredili metrike stila budući da [AssesStyle](#) nije bio od pomoći
- Pruža tabelarni prikaz pojava loših konvencija u pisanju Java koda

The screenshot shows the Eclipse IDE interface. The top part displays the source code of `Client.java` with various code elements highlighted. The bottom part shows the 'Checkstyle violations' table with 5389 markers across 25 categories.

Checkstyle violation type	Occurrences
Wrong lexicographical order for 'X' import. Should be before 'X'.	23
'X' should be separated from previous line.	3
Line is longer than X characters (found X).	97
Empty catch block.	15
Each variable declaration must be in its own statement.	1
'X' construct must use '{}'.s.	84
'X' is preceded with whitespace.	8
'X' has incorrect indentation level X, expected level should be X.	1065
'X' child has incorrect indentation level X, expected level should be one ...	3
Abbreviation in name 'X' must contain no more than 'X' consecutive ca...	8
Array brackets at illegal position.	1
Javadoc comment is placed in the wrong location.	1
Name 'X' must match pattern 'X'.	17
'X' is not preceded with whitespace.	25
Empty X block.	4
'X' at column X should have line break after.	5
Distance between variable 'X' declaration and its first usage is X, but allo...	6
'X' child has incorrect indentation level X, expected level should be X.	924
'X' is followed by whitespace.	6
'X' at column X should be on the same line as the next part of a multi-b...	20
'X' is not followed by whitespace.	189
Line contains a tab character.	2780
'X' has incorrect indentation level X, expected level should be one of th...	9
Extra separation in import group before 'X'	13
Missing a Javadoc comment.	82

- Slika 4.1. Prikaz rada CheckStyle plugin-a

- Budući da CheckStyle ne pruža automatsko ocenjivanje već samo analizu, naš tim se koristio formulom sa slajda vezan za sistem vrednovanja težine greške izračunate metrike

AssessStyle: izračunata metrika i “cenzura”

Parametar:

- Težina greške, na primer:

nedostajući komentar	4
nedostajuća prazna linija	2
loše uvlačenje	1
fajl predugačak	9
- Odnos između metrike i ocene

Ocena	Metrika
1	0 – 20
2	20 – 40
3	41 – 60
4	61 – 80
5	81 – 100
6	> 100

$$P = \sum_{f \in F} h(f) * g(f)$$

Skup svih
tipova
grešaka

Broj grešaka
tipa f u 100 LOC

Težina
greške f

- Slika 4.2. Prikaz formule računanja težine greške izračunate metrike

5. Statistic

- Pomoćna alatka koju smo koristili za detaljnu analizu LOC metrika.
- Unutar IntelliJ okruženja pruža tabelarni prikaz statistike linija koda gde prikazuje detaljno procenete zastupljenosti source linija koda, linije komentara kao i prazne linije u ukupnom LOC koda.

Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
Board.java	457	385	84%	0	0%	72	16%
LobbyController.java	370	280	76%	0	0%	90	24%
Command.java	326	280	86%	1	0%	45	14%
Client.java	313	254	81%	1	0%	58	19%
Gui.java	239	169	71%	13	5%	57	24%
GameController.java	221	168	76%	0	0%	53	24%
QueenFigure.java	205	163	80%	0	0%	42	20%
SimpleFigure.java	203	167	82%	0	0%	36	18%
Connection.java	187	147	79%	1	1%	39	21%
Server.java	173	141	82%	0	0%	32	18%
Field.java	102	40	39%	47	46%	15	15%
Field.java	95	75	79%	0	0%	20	21%
Game.java	93	69	74%	1	1%	23	25%
CheckersBoard.java	79	1	1%	61	77%	17	22%
Figure.java	56	43	77%	0	0%	13	23%
Move.java	51	40	78%	0	0%	11	22%
Player.java	32	25	78%	0	0%	7	22%
Utils.java	17	12	71%	1	6%	4	24%
FieldColor.java	16	11	69%	0	0%	5	31%
Main.java	16	13	81%	0	0%	3	19%
FigureColor.java	15	11	73%	0	0%	4	27%
GuiFigure.java	15	12	80%	0	0%	3	20%
Total:	3299	2517	76%	126	4%	656	20%

- Slika 5.1. Prikaz rada Statistic plugin-a

Rezultati metrika

1. [Projekat 1](#)
2. [Projekat 2](#)
3. [Projekat 3](#)

Projekat 1

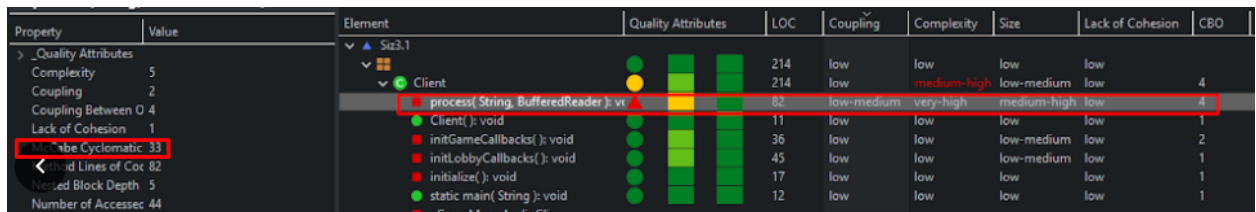
- [Ciklična složenost](#)
- [Metrika stila](#)
- [OO metrika](#)
- [LOC metrika](#)

Ciklična složenost

*Napomena: pojašnjenje grafičkog prikaza složenosti se nalazi [ovde](#)

Process unutar klase **Client.java**:

- Daje jako visoku vrednost McCabe-ove ciklične složenosti - (**33**)
Complexity - 5(very-high)
Coupling - 2(low-medium)
Ogromna količina else if grananja izaziva veliku cikličnu kompleksnost

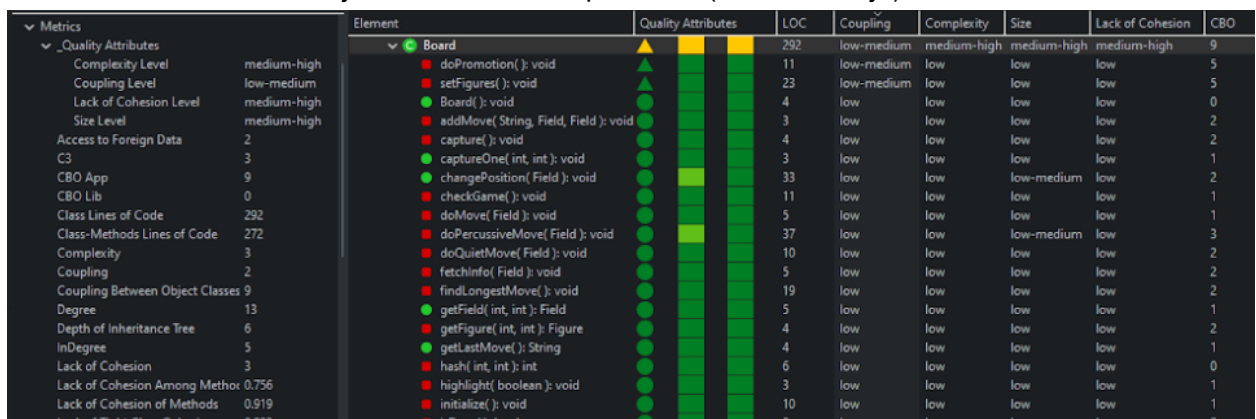


Property	Value	Element	Quality Attributes	LOC	Coupling	Complexity	Size	Lack of Cohesion	CBO
Complexity	5	Client		214	low	low	low	low	4
Coupling	2	process(String, BufferedReader): void		82	low-medium	very-high	medium-high	low	4
Coupling Between Objects	4	Client(): void		11	low	low	low	low	1
Lack of Cohesion	1	initGameCallbacks(): void		36	low	low	low-medium	low	2
M McCabe Cyclomatic	33	initLobbyCallbacks(): void		45	low	low	low-medium	low	1
Number of Lines of Code	82	initialize(): void		17	low	low	low	low	1
Number of Block Depth	5	static main(String): void		12	low	low	low	low	1
Number of Accesses	44								

- Slika 1.1. Prikaz ciklične složenosti metode process()

Klasa Board.java:

- Mnostvo metoda koje nisu međusobno povezane(niska kohezija).



Metrics	Value	Element	Quality Attributes	LOC	Coupling	Complexity	Size	Lack of Cohesion	CBO
Complexity Level	medium-high	Board		292	low-medium	medium-high	medium-high	medium-high	9
Coupling Level	low-medium	doPromotion(): void		11	low-medium	low	low	low	5
Lack of Cohesion Level	medium-high	setFigures(): void		23	low-medium	low	low	low	5
Size Level	medium-high	Board(): void		4	low	low	low	low	0
Access to Foreign Data	2	addMove(String, Field, Field): void		3	low	low	low	low	2
C3	3	capture(): void		4	low	low	low	low	2
CBO App	9	captureOne(int, int): void		3	low	low	low	low	1
CBO Lib	0	changePosition(Field): void		33	low	low	low-medium	low	2
Class Lines of Code	292	checkGame(): void		11	low	low	low	low	1
Class-Methods Lines of Code	272	doMove(Field): void		5	low	low	low	low	1
Complexity	3	doPercussiveMove(Field): void		37	low	low	low-medium	low	3
Coupling	2	doQuietMove(Field): void		10	low	low	low	low	2
Coupling Between Object Classes	9	fetchInfo(Field): void		5	low	low	low	low	2
Degree	13	findLongestMove(): void		19	low	low	low	low	2
Depth of Inheritance Tree	6	getField(int, int): Field		5	low	low	low	low	1
InDegree	5	getFigure(int, int): Figure		4	low	low	low	low	2
Lack of Cohesion	3	getLastMove(): String		4	low	low	low	low	1
Lack of Cohesion Among Methods	0.756	hash(int, int): int		6	low	low	low	low	0
Lack of Cohesion of Methods	0.919	highlight(boolean): void		3	low	low	low	low	1
		initialize(): void		10	low	low	low	low	1

- Slika 1.2. Prikaz problematičnih metoda unutar klase Board.java
- Klasa se sastoji iz mnoštva metoda koja svaka po na osobu imaju povišenu cikličnu složenost, što utiče na celokupnu sliku klase Board.java.
- Neke od metoda sa problematičnom cikličnom složenošću unutar klase Board.java su:

doPercussiveMove() - 9
setFigures() - 8
checkGame() - 7
move() - 7
findLongestMove() - 7

SimpleFigure.java:

- Visoka kompleksnost usled velikog broja grananja unutar metoda klase.

Element	Quality Attributes	LOC	Coupling	Complexity	Size
> FieldColor		7	low	low-medium	low
> Move		29	low	low	low
figures		287	low	low	low-medium
> Figure		29	low	low	low
> FigureColor		7	low	low-medium	low
> QueenFigure		123	low	low-medium	low-medium
SimpleFigure		128	low	medium-high	low-medium
getMoves(Field): List		18	low	low	low
longestRoadAboveLeft(int, int, Field, List, Li		24	low	low-medium	low
longestRoadAboveRight(int, int, Field, List, I		24	low	low-medium	low
longestRoadBellowLeft(int, int, Field, List, Li		24	low	low-medium	low
longestRoadBellowRight(int, int, Field, List, I		24	low	low-medium	low
quietMoves(Field, FigureColor): void		11	low	low	low
SimpleFigure(Image, FigureColor, Board): v		2	low	low	low

- Slika 1.3. Prikaz problematičnih metoda unutar klase SimpleFigure.java

- Klasa se sastoji iz mnoštva metoda koja svaka po na osobu imaju povišenu cikličnu složenost, što utiče na celokupnu sliku klase SimpleFigure.java.
- Metode sa problematičnom cikličnom složenošću su:
 - longestRoadAboveLeft() - 13
 - longestRoadAboveRight() - 13
 - longestRoadBellowLeft() - 13
 - longestRoadBellowRightt() - 13
 - quietMoves() - 6

Metrika stila

*Napomena: Za date ocene smo se koristili formulom koja se nalazi [ovde](#)

Skup grešaka i težinu koju nose smo definisali kao sledeće:

$F = (\text{nedostajući komentar} - 4, \text{nedostajuća prazna linija} - 2, \text{loše uvlačenje} - 1, \text{prazan blok koda} -$

Za **Client.java** ima mnoštvo grešaka u uvlačenju redova, kod if else grananja, pa je ocena 3(metrika 43), zbog lošeg uvlačenja i zbog praznih blokova.

Za **Board.java**, ima loše uvlačenje pa je ocena 3(metrika 52).

za **LobbyControler.java** isto uvlačenje predstavlja problem, pa je ocena 2(metrika 23).

za **Command.java** takođe uvlačenje predstavlja problem ali na nižoj skali, pa je ocena 1(metrika 13).

Klase **Client.java**, **GUI.java**, i **Connection.java Server.java** imaju **prazne catch blokove**, što utiče loše na metriku stila, ali ne predstavljaju neku merljivu vrednost koja bi preterano uticala na celokupnu ocenu.

OO metrika

Legenda:

- WMC - *Weighted method count*
- DIT - *Depth inheritance tree*
- LCOM - *Lack of cohesion of methods*

Paket: checkersBoard

Element	WMC	DIT	LCOM	NOC	LOC
▼ ▲ Siz3.1					
▼ 📁 checkersBoard	125				380
> 🟢 Board	98	6	0.919	0	292
> 🟢 Field	17	6	0.833	0	52
> 🟢 Move	8	1	0.81	0	29
> 🟢 FieldColor	2	2	1.333	0	7

- Slika 1.4. Prikaz OO metrike unutar paketa checkersBoard

Class: Board.java

WMC: 98 (98/125)

Komentar: S obzirom da nam je vrednost klase približno jednaka 4/5 vrednosti celog paketa, potrebno bi bilo ili razložiti na više manjih klasa ili smanjiti cikličnu kompleksnost metoda unutar klase

Classes: Board.java/Field.java

DIT: 6

Komentar: Previsoke vrednosti DIT-a nam indikuju veću dizajnersku kompleksnost klasa pošto samom dubinom u nasleđivanju, više metoda će biti uključeno u klasu. ******(DIT se preporučuje da ne bude veći od 5)

Class: FieldColor.java

LCOM: 1.333

Komentar: Visoka vrednost LCOM-a nam govori da klasa nije konherentna što nije poželjno, pošto želimo da posedujemo određen stepen enkapsulacije i zaštite.

Paket: figures

▼ 📁 figures	114				287
> 🟢 SimpleFigure	61	2	0.0	0	128
> 🟢 QueenFigure	43	2	0.0	0	123
> 🟢 Figure	8	1	0.857	2	29
> 🟢 FigureColor	2	2	1.333	0	7

-Slika 1.5. Prikaz OO metrika unutar paketa figures

Class: SimpleFigure.java

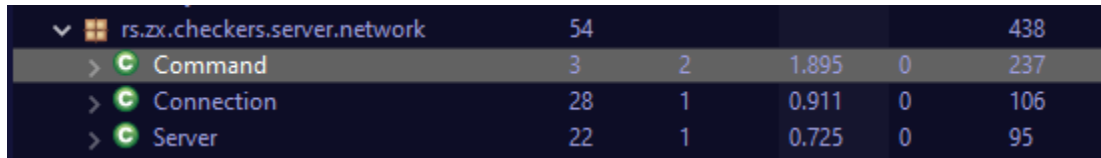
WMC: 61(61/144)

Komentar: Visoke vrednosti

Class: QueenFigure.java

WMC: 43(43/114)

Paket: **rs.zx.checkers.server.network**



rs.zx.checkers.server.network	54				438
> Command	3	2	1.895	0	237
> Connection	28	1	0.911	0	106
> Server	22	1	0.725	0	95

-Slika 1.6. Prikaz OO metrika unutar paketa rs.zx.checkers.server.network

Class: **Command.java**

LCOM: 1.895

Komentar: Iako je visoka vrednost, ne predstavlja veliki problem, zato što je ovo klasa koja je namenjena za komunikaciju sa serverom pa nam to iziskuje nisku konherentnost (zbog visoke povezanosti)

LOC metrike

Legenda:

- LOC - lines of code
- SCL - source code lines
- CL - command lines
- BL - blank lines

Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
Board.java	457	385	84%	0	0%	72	16%
LobbyController.java	370	280	76%	0	0%	90	24%
Command.java	326	280	86%	1	0%	45	14%
Client.java	313	254	81%	1	0%	58	19%
Gui.java	239	169	71%	13	5%	57	24%
GameController.java	221	168	76%	0	0%	53	24%
QueenFigure.java	205	163	80%	0	0%	42	20%
SimpleFigure.java	203	167	82%	0	0%	36	18%
Connection.java	187	147	79%	1	1%	39	21%
Server.java	173	141	82%	0	0%	32	18%
Field.java	102	40	39%	47	46%	15	15%
Field.java	95	75	79%	0	0%	20	21%
Game.java	93	69	74%	1	1%	23	25%
CheckersBoard.java	79	1	1%	61	77%	17	22%
Figure.java	56	43	77%	0	0%	13	23%
Move.java	51	40	78%	0	0%	11	22%
Player.java	32	25	78%	0	0%	7	22%
Utils.java	17	12	71%	1	6%	4	24%
FieldColor.java	16	11	69%	0	0%	5	31%
Main.java	16	13	81%	0	0%	3	19%
FigureColor.java	15	11	73%	0	0%	4	27%
GuiFigure.java	15	12	80%	0	0%	3	20%
Total:	3299	2517	76%	126	4%	656	20%

-Slika 1.7. Prikaz LOC metrika na nivou projekta

Klasa **Board.java** predstavlja klasu sa najvećim brojem ukupnih linija koda (457). Posедуje 385 linija izvornog koda što sačinjava 84% ukupnog LOC.

Iako velika u prirodi, ova klasa ima uredno napisanu strukturu kao i uredno odvajanje između metoda koristeći prazne linije(72).

U svojoj strukturi ova klasa nema linije komentara(0%) što je indikator da je kod loše dokumentovan.

Analogno klasi Board.java imamo klase kod kojih imamo isti slučaj, a to su:

LobbyController.java - LOC 370, SCL 280 (**76%**), CL- 0, BL 90 (**24%**)

Command.java - LOC 326, SCL 280 (**86%**), CL 1 (**0%**), BL 45 (**14%**)

Client.java - LOC 313, SCL 254 (**81%**), CL 1 (**0%**), BL 58 (**19%**)

Gui.java - LOC 239, SCL 169 (**71%**), CL 13 (**5%**), BL 57 (**24%**)

GameController.java - LOC 205, SCL 163 (**80%**), CL 0 (**0%**), BL 42 (**20%**)

Naredne uočene klase

agui/**Field.java** - LOC 102, CL 47 (**46%**); U skoro pola svoje strukture sadrži komentare, što nije dobra konvencija.

CheckersBoard.java -LOC 79, CL 61(**77%**); U skoro celoj svojoj strukturi je van konteksta zakomentarisana (bez ikakvog objašnjenja zašto), što uopšte nije dobra konvencija.

Naredne uočene klase

Action.java - LOC 7, SCL 4(**57%**), CL 0 (**0%**), BL 3 (**47%**)

Klasa **Action.java** u svojoj strukturi ne sadrži nikakve komentare ali zbog svoje male veličine i intuitivnosti nema potrebu za komentarima

Analogno klasi **Action.java** imamo klase kod kojih imamo isti slučaj, a to su:

ActionXY.java - LOC 6, SCL 4(**67%**), CL(**0%**), BL 2(**33%**)

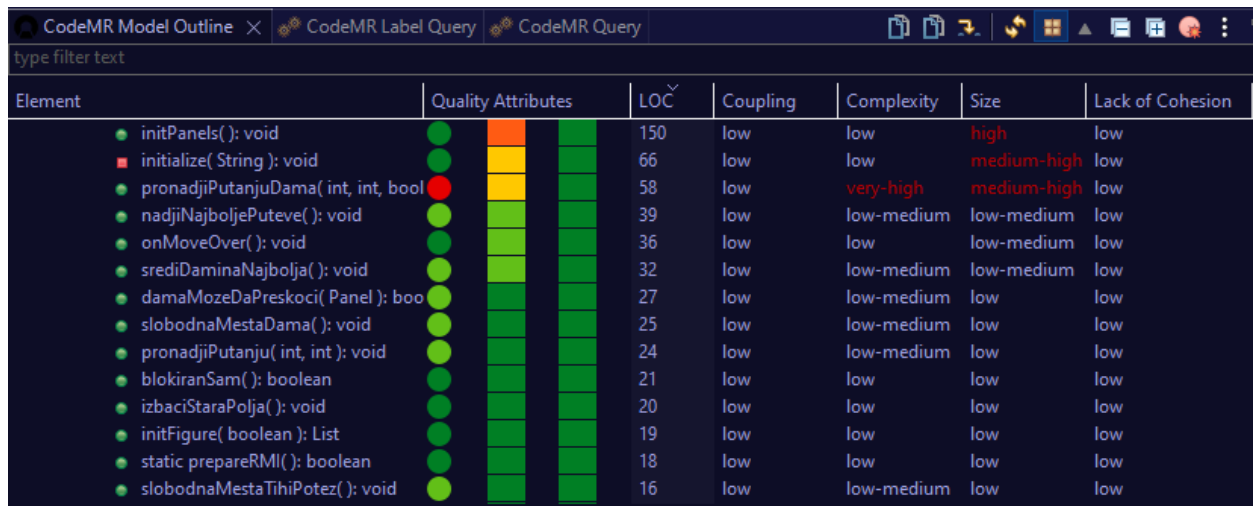
Controller.java - LOC 5, SCL 3(**60%**), CL(**0%**), BL 2(**40%**)

Projekat 2

- [Ciklična složenost](#)
- [Metrika stila](#)
- [OO metrika](#)
- [LOC metrika](#)

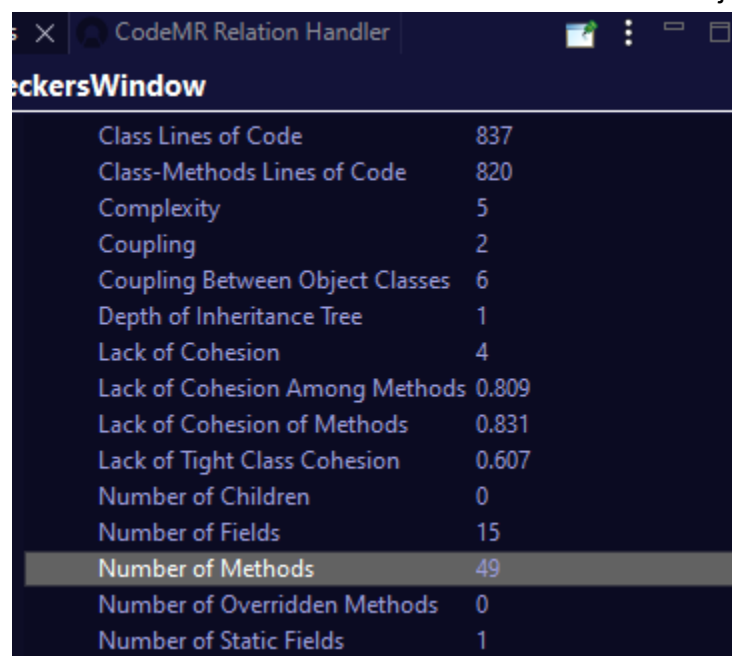
Ciklična složenost

*Napomena: pojašnjenje grafičkog prikaza složenosti se nalazi [ovde](#)



Element	Quality Attributes	LOC	Coupling	Complexity	Size	Lack of Cohesion
initPanels(): void		150	low	low	high	low
initialize(String): void		66	low	low	medium-high	low
pronadjiPutanjuDama(int, int, bool		58	low	very-high	medium-high	low
nadjiNajboljePuteve(): void		39	low	low-medium	low-medium	low
onMoveOver(): void		36	low	low	low-medium	low
srediDaminaNajbolja(): void		32	low	low-medium	low-medium	low
damaMozeDaPreskoci(Panel): boo		27	low	low-medium	low	low
slobodnaMestaDama(): void		25	low	low-medium	low	low
pronadjiPutanju(int, int): void		24	low	low-medium	low	low
blokiranSam(): boolean		21	low	low	low	low
izbaciStaraPolja(): void		20	low	low	low	low
initFigure(boolean): List		19	low	low	low	low
static prepareRMI(): boolean		18	low	low	low	low
slobodnaMestaTihiPotez(): void		16	low	low-medium	low	low

-Slika 2.1. Prikaz metoda unutar klase CheckersWindow.java



CheckersWindow	
Class Lines of Code	837
Class-Methods Lines of Code	820
Complexity	5
Coupling	2
Coupling Between Object Classes	6
Depth of Inheritance Tree	1
Lack of Cohesion	4
Lack of Cohesion Among Methods	0.809
Lack of Cohesion of Methods	0.831
Lack of Tight Class Cohesion	0.607
Number of Children	0
Number of Fields	15
Number of Methods	49
Number of Overridden Methods	0
Number of Static Fields	1

-Slika 2.2. Prikaz metrike klase CheckersWindow.java

pronadjiPutanjuDama() unutar CheckersWindow.java

- Daje jako visoku vrednost McCabe-ove ciklične složenosti (47)
- Complexity: very-high (5)
- Size: medium-high (3)
- U svojoj strukturi ima ogromnu količinu if-else grananja koja definitivno mogu da se napišu na više optimizovan način.

Ostale uočene klase imaju sličan problem s grananjem ali na manjoj skali, te klase su:

- **damaMozeDaPreskoci() (16)**
- **nadjiNajboljePuteve() (15)**
- **pronadjiPutanju() (13)**
- **slobodnaMestaDama() (17)**
- **slobodnaMestaTihiPotez () (14)**
- **srediDaminaNajbolja() (16)**

Svaka od navedenih metoda utiče znatno na celokupnu sliku ciklične složenosti klase **CheckersWindow.java**

Metrika stila

*Napomena: Za date ocene smo se koristili formulom koja se nalazi [ovde](#)

Skup grešaka i težinu koju nose smo definisali kao sledeće:

$F = (\text{nedostajući komentar} - 4, \text{loša upotreba whitespace karaktera} - 2, \text{loše uvlačenje} - 1, \text{prazan blok} - 1)$

CheckersWindow.java - nedostajuća prazna linija

ocena 3(metrika 46), loša upotreba whitespace karaktera i prazan catch blok i loše uvlačenje.

Takođe, primetili smo lošu konvenciju unutar ove klase koja predstavlja loš način imenovanja metoda, tj. neusaglašenost u davanju imena. Jedna polovina metoda je imenovana na srpskom, dok druga na engleskom.

Figura.java - loše uvlačenje.

ocena 2(metrika 27) - loše uvlačenje

Model.java - loše uvlačenje

ocena 3 (metrika 45) - loše uvlačenje.

Panel.java - loše uvlačenje

ocena 2(metrika 40) - loše uvlačenje.

RegistryManager.java - loše uvlačenje

ocena 1(metrika 13) - loše uvlačenje.

Player.java - loše uvlačenje

ocena 4(metrika 67) - loše uvlačenje.

PlayerImpl.java - loše uvlačenje

ocena 3(metrika 46) - loše uvlačenje.

Game.java - loše uvlačenje

ocena 2(metrika 30) - loše uvlačenje.

GameImpl.java - loše uvlačenje

ocena 2(metrika 27) - loše uvlačenje.

OO metrika

Legenda:

- WMC - Weighted method count
- DIT - Depth inheritance tree
- LCOM - Lack of cohesion of methods
- NOC - Number of children

Paket: GUI

Element	WMC	DIT	LCOM	NOC	LOC	Quality Attributes		
▼ ▲ Siz3.2								
▼ gui	338				1034	●	■	■
> CheckersWindow	260	1	0.831	0	837	▲	■	■
> Figura	27	5	0.773	0	86	●	■	■
> Model	17	1	0.74	0	50	●	■	■
> Panel	21	5	0.683	0	61	●	■	■

-Slika 2.3. Prikaz OO metrika paketa gui

Class: **CheckersWindow.java**

WMC: 260/338

DIT: 1

Komentar: Ova klasa zauzima najveći deo paketa, potrebno bi bilo ili razložiti na više manjih klasa ili smanjiti cikličnu kompleksnost metoda unutar klase.

Paket: GUI

Class: **Model.java**

WMC: 17/338

Komentar: Ova klasa nema grešaka.

Paket: GUI

Class: **Panel.java, Figure.java**

DIT: 5

Komentar: Ove klase bismo mogli teško ponovno da iskoristimo budući da su jako duboko ugneždene

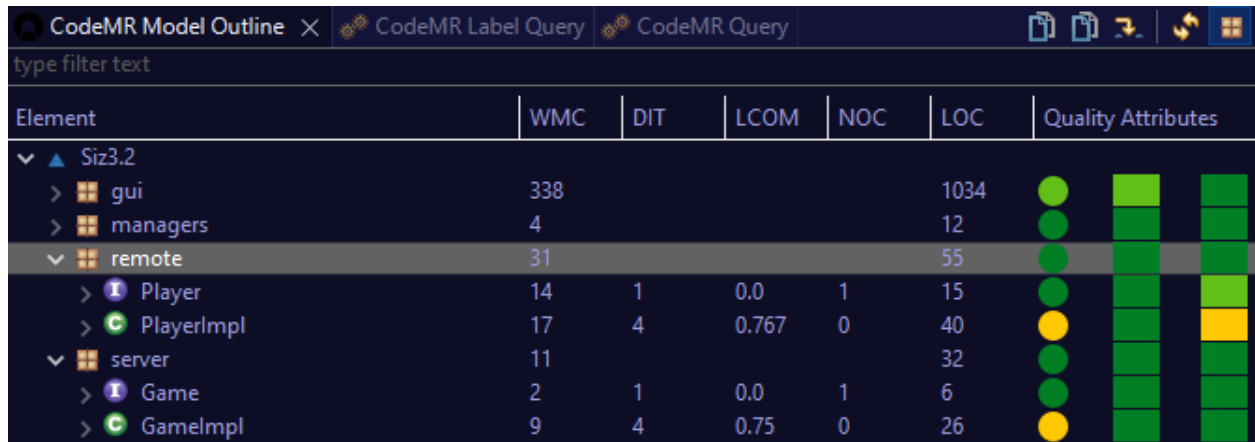
Paket: GUI

Class: **Panel.java**

LCOM = 0,683

Komentar: Visok nivo kohezije.

Paket: remote | server



Element	WMC	DIT	LCOM	NOC	LOC	Quality Attributes
▼ ▲ Siz3.2						
> gui	338				1034	● ■ ■
> managers	4				12	● ■ ■
▼ remote	31				55	● ■ ■
> Player	14	1	0.0	1	15	● ■ ■
> PlayerImpl	17	4	0.767	0	40	● ■ ■
▼ server	11				32	● ■ ■
> Game	2	1	0.0	1	6	● ■ ■
> GameImpl	9	4	0.75	0	26	● ■ ■

-Slika 2.4. Prikaz OO metrika paketa remote, server

Interface: **Player.java** | **Game.java**

NOC: 1 | 1

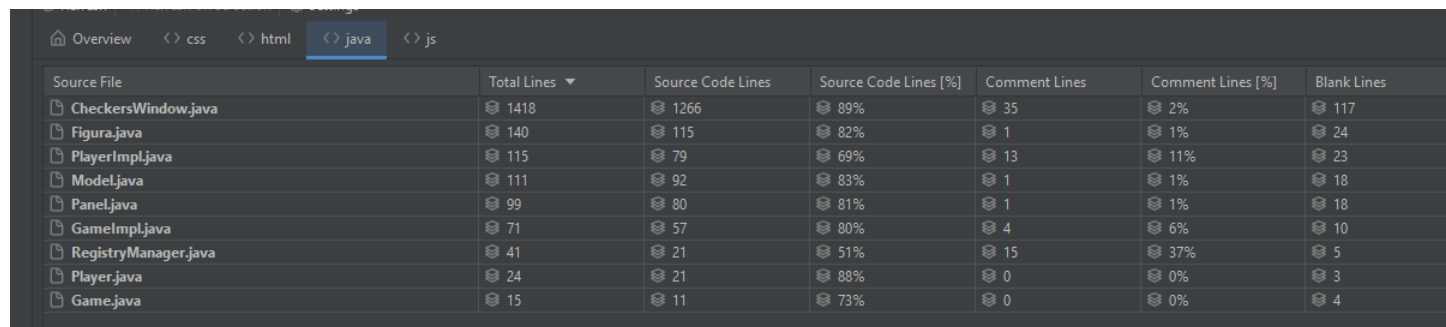
Komentar: Jedan objekat ovog interfejsa se nalazi u sklopu ovog paketa

Klase čiji se kod može lako ponovno iskoristiti: **CheckersWindow.java**, **Model.java**, **RegistryManager.java**, (interface) **Player.java**, **Game.java**

LOC metrike

Legenda:

- LOC - lines of code
- SCL - source code lines
- CL - command lines
- BL - blank lines



Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines
CheckersWindow.java	1418	1266	89%	35	2%	117
Figura.java	140	115	82%	1	1%	24
PlayerImpl.java	115	79	69%	13	11%	23
Model.java	111	92	83%	1	1%	18
Panel.java	99	80	81%	1	1%	18
GameImpl.java	71	57	80%	4	6%	10
RegistryManager.java	41	21	51%	15	37%	5
Player.java	24	21	88%	0	0%	3
Game.java	15	11	73%	0	0%	4

Slika 2.5. Prikaz LOC metrika drugog projekta

Uočene klase:

CheckersWindow.java - LOC 1418, SCL 1266(**89%**), CL 35(**2%**), BL 117(**8%**)

Klasa **CheckersWindow.java** ima enorman broj linija koda koji je jako skromno dokumentovan, gde komentari čine samo 2% ukupnog LOC što predstavlja ogroman problem za čitljivost, reusability kao i za samo razumevanje klase koja čini veliku većinu koda.

Sličan problem, na mnogo manjoj skali, imaju naredne klase:

Figura.java - LOC 140, SCL 115(**82%**), CL 1(**1%**), BL 24(**17%**)

PlayerImpl.java - LOC 115, SCL 79(**69%**), CL 13(**11%**), BL 23(**20%**)

Model.java - LOC 111, SCL 92(**83%**), CL 1(**1%**), BL 18(**16%**)

Naredne uočene klase:

Player.java - LOC 24, SCL 21(**88%**), CL 0(**0%**), BL 3(**12%**)

Klasa **Player.java** u svojoj strukturi ne sadrži nikakve komentare ali zbog svoje male veličine i intuitivnosti nema potrebu za komentarima

Analogno klasi **Player.java** imamo klasu kod kojih imamo isti slučaj, a to je:

Game.java - LOC 15, SCL 11(**73%**), CL 0(**0%**), BL 4(**27%**)

Projekat 3

- [Ciklična složenost](#)
- [Metrika stila](#)
- [OO metrika](#)
- [LOC metrika](#)

Ciklična složenost

*Napomena: pojašnjenje grafičkog prikaza složenosti se nalazi [ovde](#)

CheckersTable.java

	Element	Quality Attributes	LOC	Coupling	Complexity
CBO Lib	0				
Class Lines of Code	126				
Class-Methods Lines of Code	120				
Complexity	2				
Coupling	2				
Coupling Between Object Classes	7				
Depth of Inheritance Tree	1				
Lack of Cohesion	3				
Lack of Cohesion Among Methods	0.743				
Lack of Cohesion of Methods	0.743				
Lack of Tight Class Cohesion	0.59				
Number of Children	0				
Number of Fields	5				
Number of Methods	17				
Number of Overridden Methods	0				
	▼ CheckersTable		126	low-medium	low-medium
	■ checkAvailableJumpForFigure(int, i		12	low-medium	low
	■ checkAvailableMoveForFigure(int,		14	low-medium	low
	● checkAvailableMovesForQueen(int		6	low	low
	● checkMovesForPlayer(int): void		18	low	low
	● getAvailableJumpsMap(): Map		3	low	low
	● getAvailableMovesMap(): Map		3	low	low
	● getField(int, int): Field		2	low	low
	● getFieldCoordinates(Integer): Coo		6	low	low
	● getTable(): Field		2	low	low
	● move(Integer, Integer): void		11	low	low
	● move(Integer, JumpCouple): void		6	low	low
	● notOutOfRange(int, int): boolean		2	low	low
	● printTable(): void		14	low	low

-Slika 3.1. Prikaz metoda unutar klase CheckersTable.java, kao i prikaz metrika iste klase

- checkAvailableMoveForFigure() (8)

Ova metoda u svom zaglavlju ima ugneždjena if grananja koja će se proveravati svakim pozivom ove metode, što izaziva visoku cikličnu složenost.

MainFrame.java

	Element	Quality Attributes	LOC	Coupling	Complexity	Size
rs.gui.MainFrame	▼ MainFrame		393	medium-high	high	medium-high
	> Lcheckers/gui/MainFrame\$3433;		0	low	low	low
	> Lcheckers/gui/MainFrame\$4444;		0	low	low	low
	> Listener		21	low	low	low
	● deselectFields(Field): void		30	low-medium	low-medium	low
	● MainFrame(UserInfo): void		72	low-medium	low	medium-high
	● actionPerformed(ActionEvent): vo		24	low	low	low
	● gameOver(): void		8	low	low	low
	● initMainArea(): void		31	low	low	low-medium
	● isBusy(Player): void		6	low	low	low
	● isOpponentBusy(): void		6	low	low	low
	■ log(int, int, boolean): void		20	low	low	low
	● markAllowedField(): void		14	low-medium	low	low
	● markAllowedFieldForMoves(Field)		24	low-medium	low	low
	● onExit(): void		3	low	low	low
	● onMove(Integer, Integer, boolean)		20	low	low	low
	● onNo(): void		3	low	low	low
	● onRequest(String, Player): void		15	low-medium	low	low
	● onWin(): void		3	low	low	low
	● onYes(): void		9	low	low	low
	■ request(String): int		3	low	low	low
	■ restoref(): void		6	low	low	low

-Slika 3.1. Prikaz metoda unutar klase MainFrame.java kao i metrika iste klase

- deselectFields() (13)

Ova Metoda unutar svog tela sadrži enormnu količinu if-else grananja, kao i veliku količinu for petlji koje znatno utiču na cikličnu složenost klase u kojoj se nalazi.

- **actionPerformed() (8)**

Ova metoda takođe unutar svog tela sadrži znatnu količinu if-else grananja ali i veliku količinu try catch blokova koji zajedno utiču na povišenu cikličnu složenost ove metode.

- **markAllowedFieldForMoves() (9)**

Ova metoda sadrži veliki broj ugneždenih if grananja koji unutar svog tela sadrže for petlje pa isto tako znatno utiču na cikličnu kompleksnost ove metode.

Metrika stila

*Napomena: Za date ocene smo se koristili formulom koja se nalazi [ovde](#)

Skup grešaka i težinu koju nose smo definisali kao sledeće:

$F = (\text{prazno telo} - 2, \text{loša upotreba whitespace karaktera} - 2, \text{loše uvlačenje} - 1, \text{prazan blok koda} -$

CheckersTable.java - loše uvlačenje
ocena 1(metrika 13) - loše uvlačenje.

Field.java -loše uvlačenje
ocena 3(metrika 44) - loše uvlačenje i prazno telo.

Figure.java - loše uvlačenje
ocena 1(metrika 18) - loše uvlačenje

JumpCouple.java -loše uvlačenje
ocena 3(metrika 44) - loše uvlačenje.

JumpMove.java - loše uvlačenje
ocena 2(metrika 30) - loše uvlačenje

Move.java -loše uvlačenje
ocena 3(metrika 42) - loše uvlačenje

Login.java - loše uvlačenje
ocena 2(metrika 38) - loše uvlačenje, catch blok, prazno telo {}, loša upotreba whitespace.

MainFrame.java - loše uvlačenje
ocena 2(metrika 25) -loše uvlačenje i prazno telo.

Table.java - loše uvlačenje
ocena 3 (metrika 47) - loše uvlačenje.

UsersTableModel.java -loše uvlačenje
ocena 2(metrika 27) - loše uvlačenje

Player.java -loše uvlačenje
ocena 2(metrika 21) - loše uvlačenje

PlayerImpl.java - loše uvlačenje
ocena 1(metrika 15) - loše uvlačenje

RegistryManager.java - loše uvlačenje
ocena 1(metrika 11) -loše uvlačenje

UserImplementation.java - loše uvlačenje
ocena 2(metrika 31) - loše uvlačenje

UserInfo.java - loše uvlačenje
ocena 3(metrika 51) - loše uvlačenje, prazno telo, prazna dokumentacija.

OO metrika

Legenda:

- WMC - *Weighted method count*
- DIT - *Depth inheritance tree*
- LCOM - *Lack of cohesion of methods*
- NOC - *Number of children*

Paket: checkers.gui

Element	WMC	DIT	LCOM	LOC	NOC
▼ ▲ Siz3.3					
▼ checkers.gui	134			636	
> C MainFrame	77	6	0.9	393	0
> C Table	19	5	0.841	74	0
> C UsersTableModel	14	2	0.7	56	0
> C Login	7	6	0.857	113	0

-Slika 3.2. Prikaz OO metrika paketa checkers.gui

Class: **MainFrame.java**

WMC: 77/134

DIT : 6

Komentar: sa sigurnošću možemo da kažemo da može da se razloži na manje celine, isto tako ima dubinu nasleđivanja 6 koja prelazi neku gornju granicu od 5.

Paket: **checkers.entity**

▼ checkers.entity	81			229	
> C CheckersTable	47	1	0.743	126	0
> C Field	8	5	0.813	30	0
> C Figure	7	1	0.792	21	0
> C Move	6	1	0.5	15	1
> C Coordinates	5	1	0.5	14	0
> C JumpCouple	5	1	0.5	14	0
> C JumpMove	3	2	0.0	9	0

-Slika 3.3. Prikaz OO metrika paketa checkers.entity

Komentar: Na nivou paketa dubina nasleđivanja je pretežno 1, što nam je indikator da nije dovoljno dobro iskorišćen koncept OO dizajna, analogno tome je situacija i kod paketa **checkers.server**

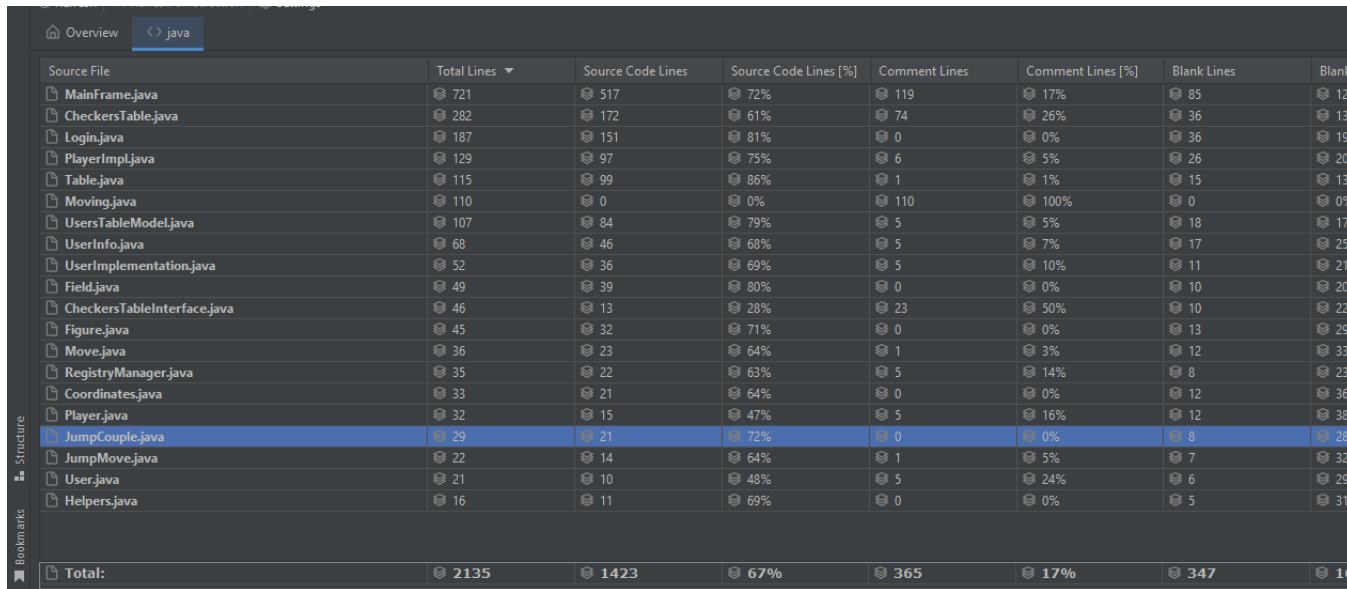
▼ checkers.server	26			75	
> C UserInfo	13	1	0.733	36	0
> C UserImplementation	6	4	0.667	21	0
> C RegistryManager	4	1	0.0	13	0
> I User	3	1	0.0	5	1

-Slika 3.4. Prikaz OO metrika paketa checkers.server

LOC metrike

Legenda:

- LOC - lines of code
- SCL - source code lines
- CL - comment lines
- BL - blank lines



Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
MainFrame.java	721	517	72%	119	17%	85	12%
CheckersTable.java	282	172	61%	74	26%	36	13%
Login.java	187	151	81%	0	0%	36	19%
PlayerImpl.java	129	97	75%	6	5%	26	20%
Table.java	115	99	86%	1	1%	15	13%
Moving.java	110	0	0%	110	100%	0	0%
UsersTableModel.java	107	84	79%	5	5%	18	17%
UserInfo.java	68	46	68%	5	7%	17	25%
UserImplementation.java	52	36	69%	5	10%	11	21%
Field.java	49	39	80%	0	0%	10	20%
CheckersTableInterface.java	46	13	28%	23	50%	10	22%
Figure.java	45	32	71%	0	0%	13	29%
Move.java	36	23	64%	1	3%	12	33%
RegistryManager.java	35	22	63%	5	14%	8	23%
Coordinates.java	33	21	64%	0	0%	12	36%
Player.java	32	15	47%	5	16%	12	38%
JumpCouple.java	29	21	72%	0	0%	8	28%
JumpMove.java	22	14	64%	1	5%	7	32%
User.java	21	10	48%	5	24%	6	29%
Helpers.java	16	11	69%	0	0%	5	31%
Total:	2135	1423	67%	365	17%	347	16%

Slika 3.5. Prikaz LOC metrike treceg projekta

MainFrame.java - LOC 721, SCL 517(72%), CL 119(17%), BL 85(12%)

Klasa **MainFrame.java** je u svojoj strukturi glomazna ali baš zbog svoje prirode i mora da bude takva budući da je to kod za JavaFX (gui elemente) koji ne može drugačije(kraće) da se zapiše.

Klasa na ovoj skali veličine deluje sasvim dobro dokumentovano sa zdravih 119 linija komentara.

Analogno klasi **MainFrame.java** imamo klasu koja je, na manjoj skali, u sličnoj situaciji:

CheckersTable.java - LOC 282, SCL 172(61%), CL 74(26%), BL 36(13%)

Naredne uočene klase:

Login.java - LOC 187, SCL 151(81%), CL 0(0%), BL 36(19%)

Klasa **Login.java** u svojoj strukturi nema komentara, međutim za time i nema potrebe budući da deluje kao klasa koja je izgenerisana koristeći JavaSwing

PlayerImpl.java - LOC 129, SCL 97(**75%**), CL 6(**5%**), BL 26(**20%**)

Klasa **PlayerImpl.java** u svojoj strukturi nema komentara, što predstavlja problem za čitljivost i razumevanje napisane klase.

Analogno prethodno navedenoj klasi, imamo **Table.java** - LOC 115, SCL 99(**86%**), CL 1(**1%**), BL 15(**13%**)

Naredne uočene klase:

Moving.java - LOC 100, SCL 0(**0%**), CL 100(**100%**), BL 0(**0%**)

Klasa **Moving.java** je u svojoj celosti zakomentarisana pa možemo sa sigurnošću da kažemo da joj u ovom projektu nije mesto.

CheckersTableInterface.java - LOC 46, SCL 13(**28%**), CL 23(**50%**), BL 10(**22%**)

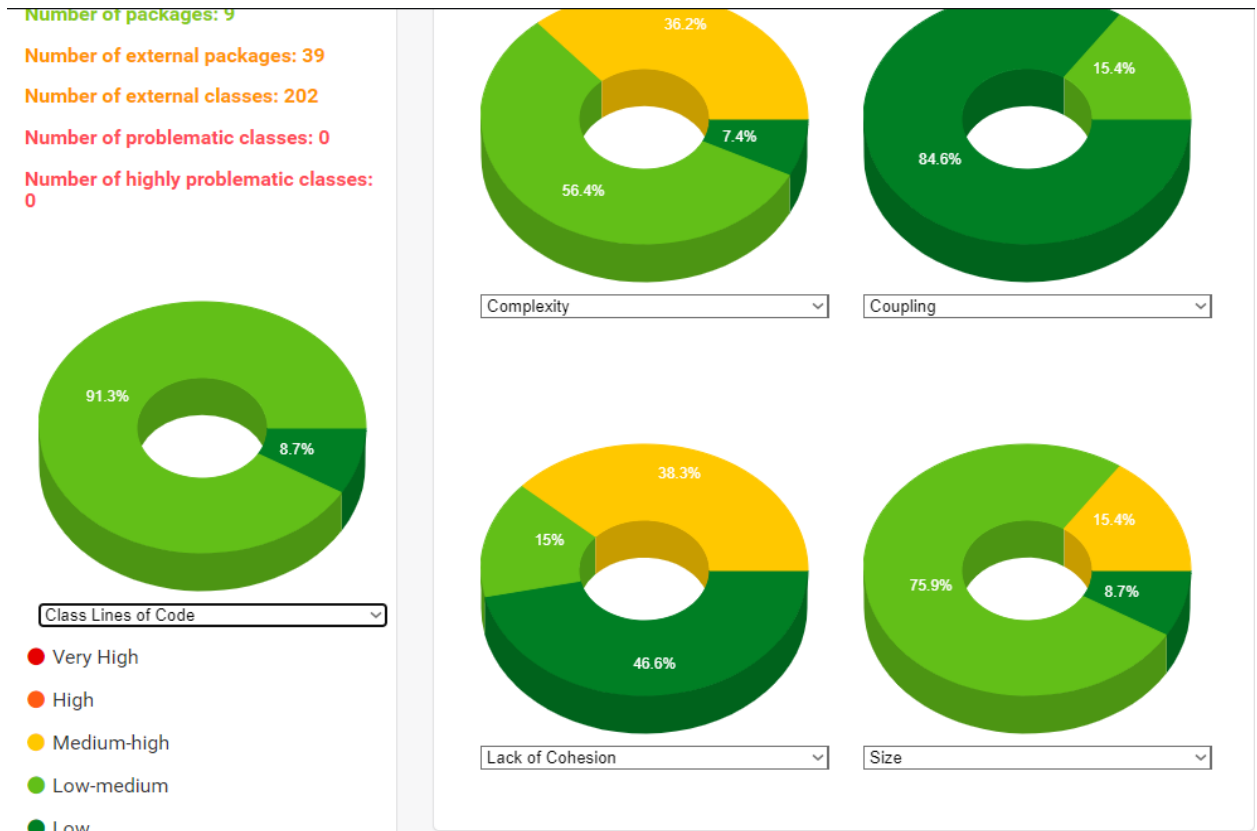
Klasa **CheckersTableInterface.java** je u svojoj strukturi napola zakomentarisana što indikuje detaljnu dokumentaciju metoda što bi značilo da bi te iste metode mogle da se optimizuju ili napišu na više razuman način.

Helpers.java - LOC 16, SCL 11(**69%**), CL 0(**0%**), BL 5(**31%**)

Klasa **Helpers.java** u svojoj strukturi ne poseduje nikakav komentar ali za njega i nema potrebe budući da je klasa napisana kratko, čitko i razumljivo. Jedina loša stavka ove klase jeste to što ima veliki broj praznih linija, što razvlači kod kod previše i čini ga manje čitkim.

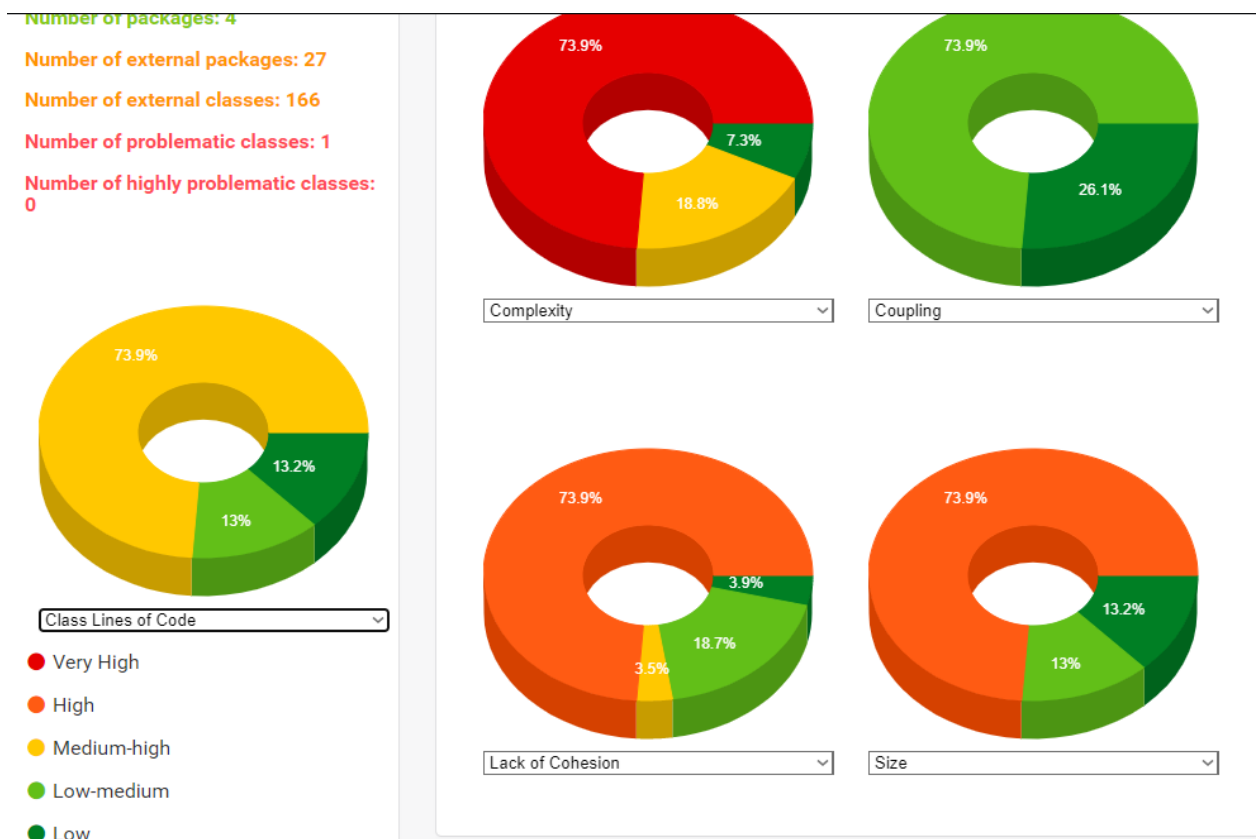
Analogno klasi **Helpers.java**, imamo identičnu situaciju s klasom **JumpCouple.java** - LOC 29, SCL 21(**72%**), CL 0(**0%**), BL 8(**28%**)

Analiza i poređenja rezultata



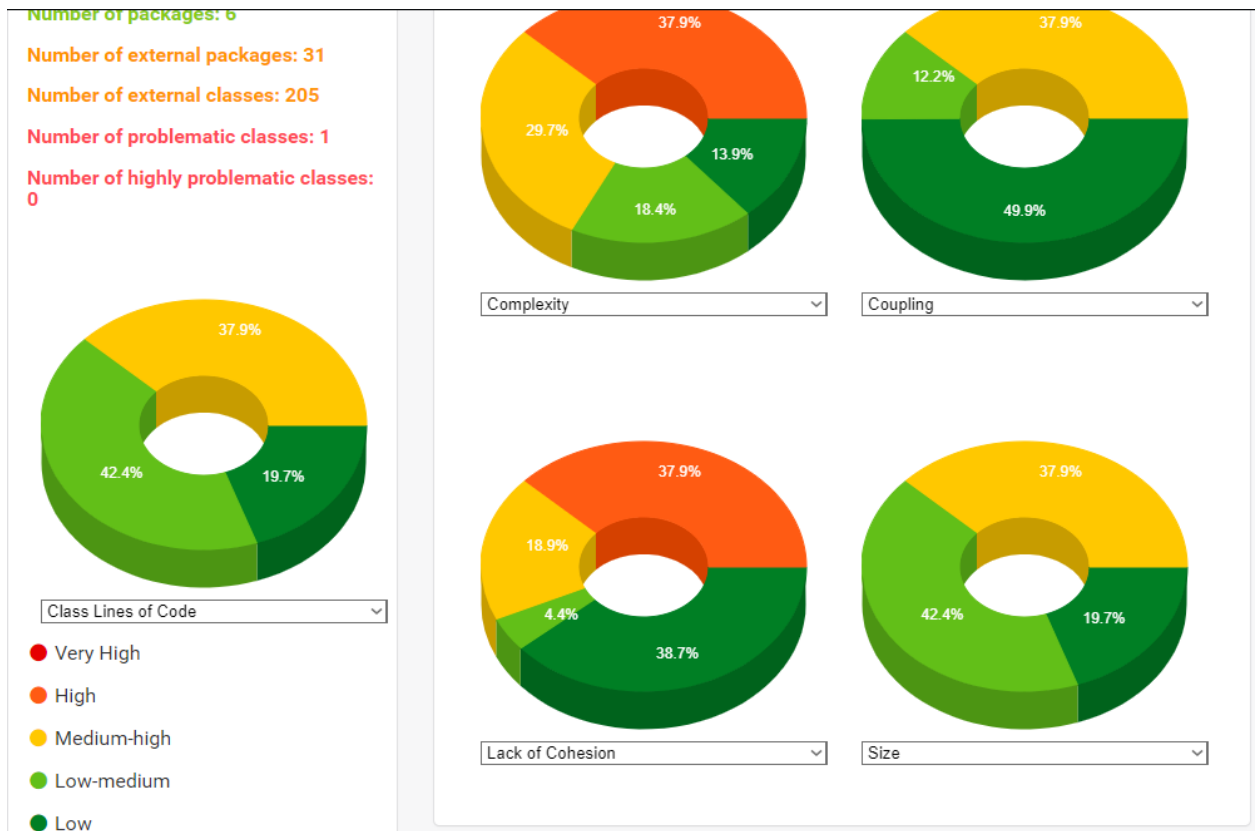
Slika 4.1. Statistički prikaz metrika prvog projekta

- Pregledom i analizom svih izmerenih metrika unutar prvog projektnog zadatka primetili smo da su koncepti OO dizajna dobro primenjeni na nivou celokupnog projekta. Taj zaključak donosimo iz niske izmerene ciklične složenosti većine metoda i ne preterano dubokog nivoa nasleđivanja između klasa. Takođe, manjak kohezije je nizak na nivou svih paketa unutar projekta. Metrike stila su indikovale da su se konvencije dokumentovanja koda mogle više poštovati, budući da smo imali određen broj loše dokumentovanih klasa tj. metoda unutar istih.



Slika 4.1. Statistički prikaz metrika drugog projekta

- Za razliku od prvog projektnog zadatka, drugi zadatak nije poštovao principe OO dizajna, gde imamo visok manjak kohezije među metodama klase, što indikuje da nam većina klase sadrže enorman broj metoda koje nemaju neku međusobnu povezanost niti zavisnost. Ciklična složenost je jako visoka usled loše implementiranih i napisanih metoda. Takođe, pojedine klase su bespotrebno ogromne u veličini, gde sadrže preko 700 linija koda koje su pritom loše dokumentovane.



Slika 4.1. Statistički prikaz metrika trećeg projekta

- Treći projektni zadatak deluje kao da je uzeo sve dobro iz prvog i iz drugog zadatka i spojio u jednu funkcionalnu i koherentnu strukturu, gde su se poštovale OO metrike (samo jedan slučaj gde je manjak kohezije visok) i gde je kod sasvim dovoljno dokumentovan. Struktura koda kao takvog je isto tako dobro izgrađena, gde ne postoji mnoštvo klasa sa po preko 500 linija koda, već su sve podeljene u manje logičke celine. Ciklična složenost u ovom projektu ne predstavlja veliku problematiku budući da je jako mali broj problematičnih metoda, gde metoda sa najvišom metrikom iznosi 13.

Zaključak

Nakon detaljne analize i poređenja rezultata svake od izmerenih metrika, zaključili smo da je treći projektni zadatak najbolje implementiran, budući da su u tom zadatku ispoštovane sve izmerene metrike na više nego prihvatljivom nivou.

Dalje, druga najbolja implementacija predstavlja prvi projektni zadatak. Iako u u svojoj implementaciji manjka u metrikama stila, i dalje poštuje sve ostale izmerene metrike.

Kao najgoru implementaciju, predlažemo drugi projektni zadatak. Manjka u oblasti OO metrike, ciklična komplektnost mu je *grozna*, a i metrike stila nisu uopšte ispoštovane. Klase su bespotrebno jako dugačke, sa velikim manjkom kohezije i sa jako malo prpratne dokumentacije.

Organizazione beleške

- [Sastanak 0](#)
- [Sastanak 1](#)
- [Sastanak 2](#)
- [Sastanak 3](#)
- [Sastanak 4](#)
- [Sastanak 5](#)
- [Sastanak 6](#)

-

Sastanak 0

Sastanak održan 10.01.2022. u 19h

Sastanak je trajao 30 min

Svi učesnici su prisustvovali sastanku

Svi pristupi su aktivno učestvovali

Određen je plan izrade projekta

Određena je agenda za naredni sastanak

Sastanak 1

Sastanak održan 11.01.2022. u 19h

Sastanak je trajao 1h 40 min

Većina učesnika je aktivno učestvovala(4/6) u izradi projekta

Svi učesnici su aktivno učestovali

Određene su ciklične metrike unutar prvog projekta

Određena je agenda za naredni sastanak

Sastanak 2

Sastanak održan 12.01.2022. u 19h

Sastanak je trajao 2h 15 min

Jedan učesnik je izostao

Tri od pet prisutnih učesnika su aktivno učestvovali u izradi projekta

Određene su metrike stila za sva tri projekta

Započeto je određivanje metrika linija koda za prvi projekat

Određena je agenda za naredni sastanak

Sastanak 3

- Sastanak održan 13.01.2022. u 19h
- Sastanak je trajao 2h 15 min
- Dva učesnika su izostala
- Svi učesnici su aktivno učestvovali u izradi projekta
- Određene su OO metrike za sva tri projekta
- Određena je agenda za naredni sastanak

Sastanak 4

- Sastanak održan 14.01.2022. u 19h
- Sastanak je trajao 2 h 10 min
- Jedan učesnik je izostao (kasnije 2)
- Skoro svi učesnici su aktivno učestvovali(4/5), kasnije svi(4/4)
- Odrađene su LOC metrike za sva tri projekta
- Učesnica Željana Pujin je naprasno napustila sastanak dok je trajao bez ikakve napomene, na istom je prisustvovala isključivo fizički - ceo sastanak je bila mutirana (kao i na svim prethodnim sastancima vezanim za treći praktični zadatak)
- Na posletku sastanka i po dobijanju rezultata poena drugog praktičnog zadatka, koleginica je pitala tim zašto je ocenjena kako jeste (7 od maksimalnih 8 poena koji je ostvario naš tim), na šta je dobila odgovor od jednog učesnika "Verovatno shodno uloženom trudu" (Uroš Đerić) i ubrzo nakon toga je napustila grupni čet na platformi Messenger.
- Nakon toga šef tima (Uroš Đerić) je pokušao da kontaktira koleginicu privatno i dobio odgovor da koleginica Željana ne planira da dalje učestvuje u izradi projekta naglasivši da je za nju projekat završen, kako ona kaže.
- Određena je agenda za naredni sastanak

Sastanak 5

- Sastanak održan 15.01.2022. u 19h
- Sastanak je trajao 2 h 15 min
- Jedan učesnik je izostao
- Svi učesnici su aktivno učestvovali
- Ispisani su i uredno unešeni rezultati svih metrika za prvi projekat
- Određena je agenda za finalni sastanak

Sastanak 6

- Sastanak održan 16.01.2022. u 15h
- Sastanak je trajao 2 h 15 min
- Jedan učesnik je izostao
- Svi prisutni učesnici su aktivno učestvovali
- Ispisani su i uredno unešeni rezultati metrika za preostale projektne zadatke
- Projekat je završen!

Izvori korišćenih resursa

<https://marketplace.eclipse.org/content/codemr-static-code-analyser>

<http://metrics2.sourceforge.net>

<https://perun.pmf.uns.ac.rs/java/workshops/AssessStyle.zip>

<https://checkstyle.org/eclipse-cs/>

<https://app.printscr.com/en/index.html>

<https://www.eclipse.org>

<https://www.jetbrains.com/idea/>

<https://plugins.jetbrains.com/plugin/4509-statistic>