LOCATING THE ARCHITECTURAL ROOTS OF TECHNICAL DEBT

R. KAZMAN, Y. CAI, R. MO, Q. FENG,
L. XIAO, S. HAZIYEV, V. FEDAK,
A. SHAPOCHKA

YOUR TYPICAL SOFTWARE PROJECT

The boat is leaking but you keep paddling!

Why?

1. **The illusion of progress.**
2. **The lack of measurements.**

TECHNICAL DEBT

The state of the practice in "technical debt" or "code smell" identification: informal, experience- and intuition-based.

"Debt" is still largely a metaphor.

But software managers must make decisions on a *financial* basis.

How do we bridge this gap?

WHAT WE DID

Case study with SoftServe Inc. in an attempt to quantify

1. **the cost of technical (architecture) debt and**
2. **the benefits of repairing the debt.**

WHAT WE DID

- **Used the Design Rule Space (DRSpace) analysis approach* to locate architecture debts**
- **Visualized the architecture flaws in these DRSpaces using our tool Titan**
- **Extracted project data to quantify the penalty these debts were incurring**
- **Estimated the potential benefits of refactoring**
- **Made a business case to justify refactoring.**

*L. Xiao, Y. Cai, R. Kazman, "Design Rule Spaces: A New Form of Architecture Insight", *Proceedings of ICSE 2014,* June 2014.

ARCHITECTURAL FLAWS

BRIDGING THE GAP BETWEEN ARCHITECTURE AND QUALITY

Using Titan we can find architecture design flaws:*

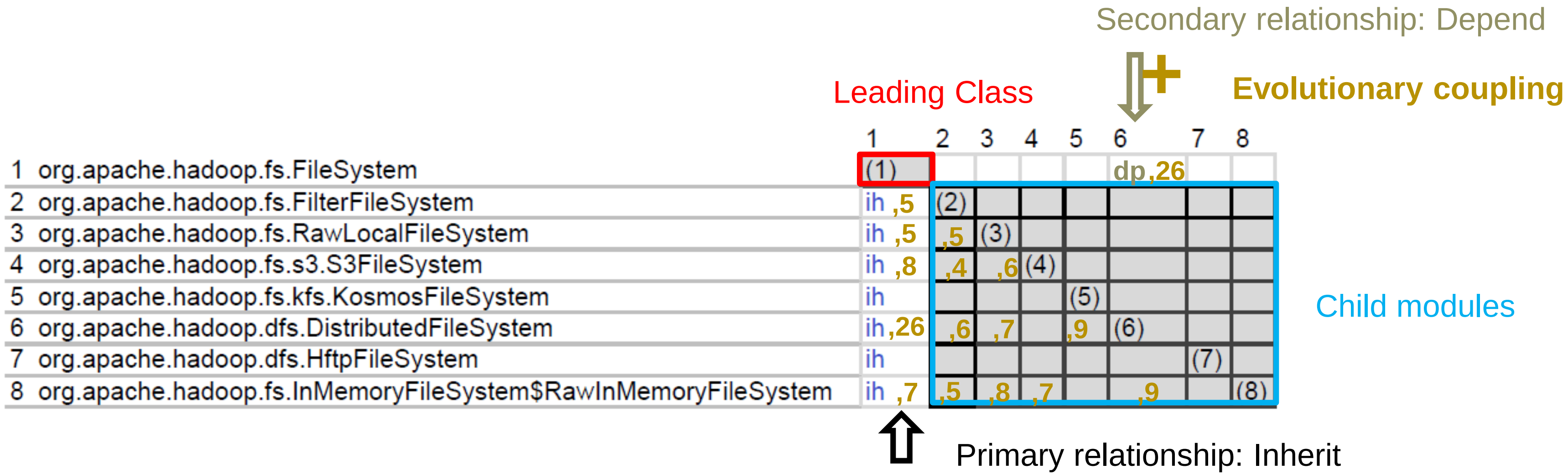
- **cyclic class dependencies**
- **cyclic package dependencies**
- **improper inheritance**
- **modularity violations**
- **unstable interfaces**

Identifying these flaws allows us to:

- **Locate and assess technical debt and its economic impact**
- **predict the economic impact of repair strategies**

*R. Mo, Y. Cai, R. Kazman, L. Xiao, “Hotspot Patterns: The Formal Definition and Automatic Detection of Architecture Smells”, *Proceedings of WICSA 2015*, May 2015.

EXAMPLE FLAWS



Secondary relationship: Depend

Evolutionary coupling

Leading Class

		1	2	3	4	5	6	7	8
1	org.apache.hadoop.fs.FileSystem	(1)					dp,26		
2	org.apache.hadoop.fs.FilterFileSystem	ih ,5	(2)						
3	org.apache.hadoop.fs.RawLocalFileSystem	ih ,5	,5	(3)					
4	org.apache.hadoop.fs.s3.S3FileSystem	ih ,8	,4	,6	(4)				
5	org.apache.hadoop.fs.kfs.KosmosFileSystem	ih				(5)			
6	org.apache.hadoop.dfs.DistributedFileSystem	ih,26	,6	,7		,9	(6)		
7	org.apache.hadoop.dfs.HftpFileSystem	ih						(7)	
8	org.apache.hadoop.fs.InMemoryFileSystem$RawInMemoryFileSystem	ih ,7	,5	,8	,7		,9		(8)

Child modules

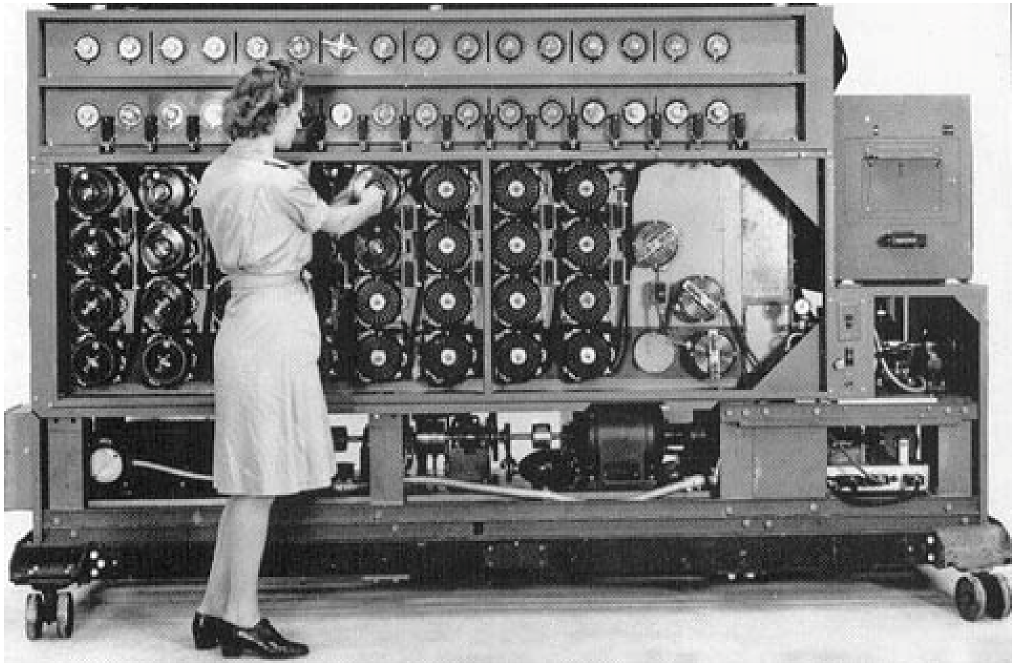
Primary relationship: Inherit

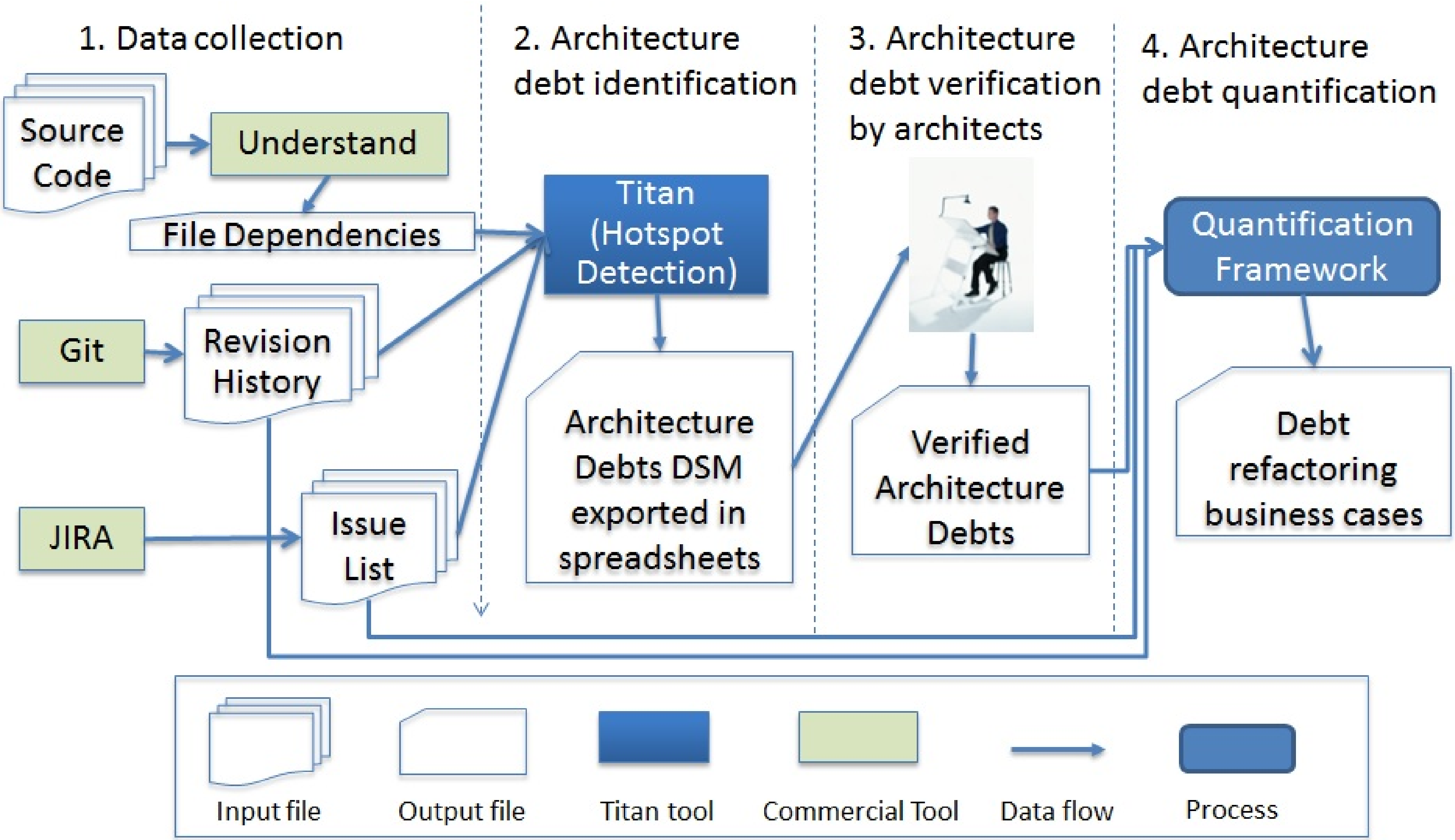
Issue 1: Parent class depends on child class

Issue 2: Unusual evolutionary coupling between parent and child class

Issue 3: Modularity violation

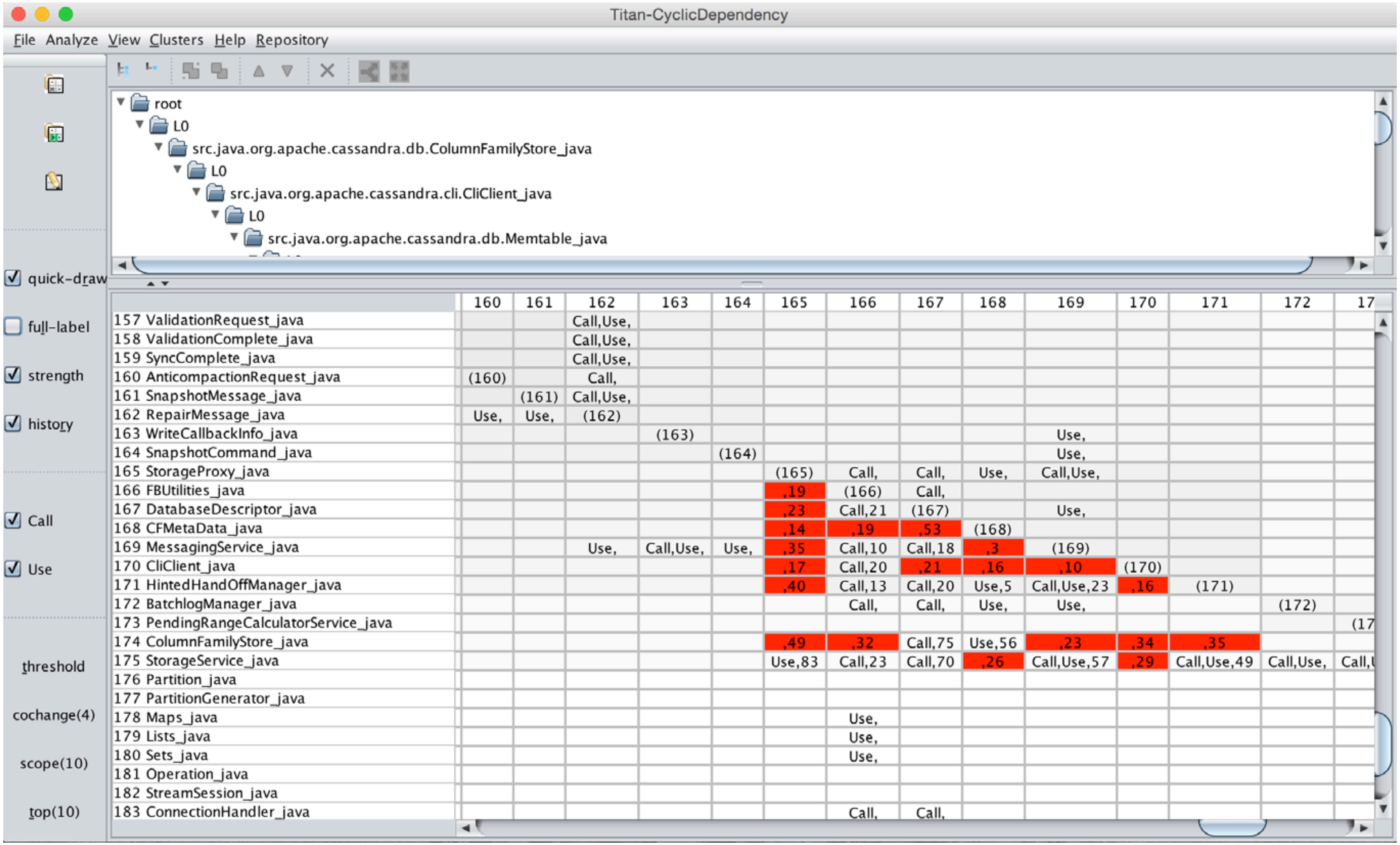
HOW DO WE COMPUTE DRSPACES AND ARCHITECTURE FLAWS?




1. Data collection
2. Architecture debt identification
3. Architecture debt verification by architects
4. Architecture debt quantification
Source Code
Understand
File Dependencies
Git
Revision History
JIRA
Issue List
Titan (Hotspot Detection)
Architecture Debts DSM exported in spreadsheets
Verified Architecture Debts
Quantification Framework
Debt refactoring business cases
Input file
Output file
Titan tool
Commercial Tool
Data flow
Process

*L. Xiao, Y. Cai, R. Kazman, “Titan: A Toolset That Connects Software Architecture with Quality Analysis”, *Proceedings of FSE 2014*, (Hong Kong), November 2014.

TITAN GUI



- root
  - L0
    - src.java.org.apache.cassandra.db.ColumnFamilyStore_java
      - L0
        - src.java.org.apache.cassandra.cli.CliClient_java
          - L0
            - src.java.org.apache.cassandra.db.Memtable_java

☑ quick-draw

□ full-label

☑ strength

☑ history

☑ Call

☑ Use

threshold

cochange(4)

scope(10)

top(10)

	160	161	162	163	164	165	166	167	168	169	170	171	172	17
157 ValidationRequest_java			Call,Use,											
158 ValidationComplete_java			Call,Use,											
159 SyncComplete_java			Call,Use,											
160 AnticompactionRequest_java	(160)		Call,											
161 SnapshotMessage_java		(161)	Call,Use,											
162 RepairMessage_java	Use,	Use,	(162)											
163 WriteCallbackInfo_java				(163)						Use,				
164 SnapshotCommand_java					(164)					Use,				
165 StorageProxy_java						(165)	Call,	Call,	Use,	Call,Use,				
166 FBUtilities_java						,19	(166)	Call,						
167 DatabaseDescriptor_java						,23	Call,21	(167)		Use,				
168 CFMetaData_java						,14	,19	,53	(168)					
169 MessagingService_java			Use,	Call,Use,	Use,	,35	Call,10	Call,18	,3	(169)				
170 CliClient_java						,17	Call,20	,21	,16	,10	(170)			
171 HintedHandOffManager_java						,40	Call,13	Call,20	Use,5	Call,Use,23	,16	(171)		
172 BatchlogManager_java							Call,	Call,	Use,	Use,			(172)	
173 PendingRangeCalculatorService_java														(17
174 ColumnFamilyStore_java						,49	,32	Call,75	Use,56	,23	,34	,35		
175 StorageService_java						Use,83	Call,23	Call,70	,26	Call,Use,57	,29	Call,Use,49	Call,Use,	Call,
176 Partition_java														
177 PartitionGenerator_java														
178 Maps_java							Use,							
179 Lists_java							Use,							
180 Sets_java							Use,							
181 Operation_java														
182 StreamSession_java														
183 ConnectionHandler_java							Call,	Call,						

PRIOR RESEARCH RESULTS

RQ1: A significant portion of the DRSpaces led by an error prone class are also error-prone.

RQ2: The 5 largest DRSpaces always captured more than half of the buggy files in the project.

RQ3: Error-prone DRSpaces have structural problems and modularity violations.

RQ4: If a file is involved in greater numbers of architecture issues, it is more error-prone/change-prone than average files.

ECONOMIC ANALYSIS

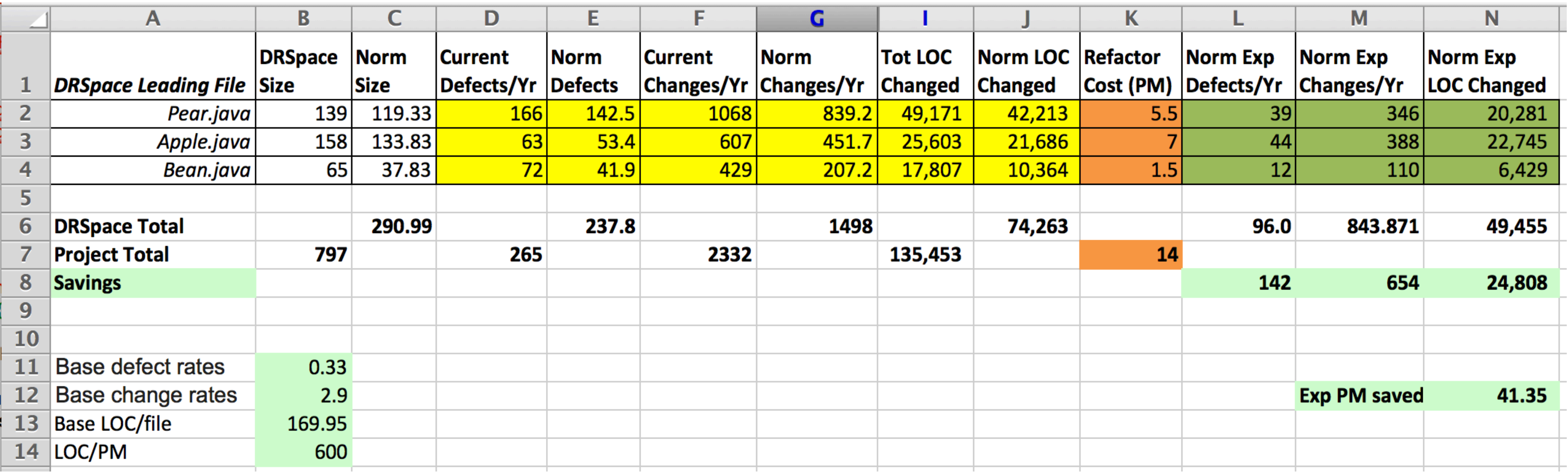
Based on the identified DRSpaces and an identification of their architecture flaws, we can plan refactoring strategies.

And we can make decisions about *whether* to refactor based on ROI.

This analysis is entirely based on commonly available project data.

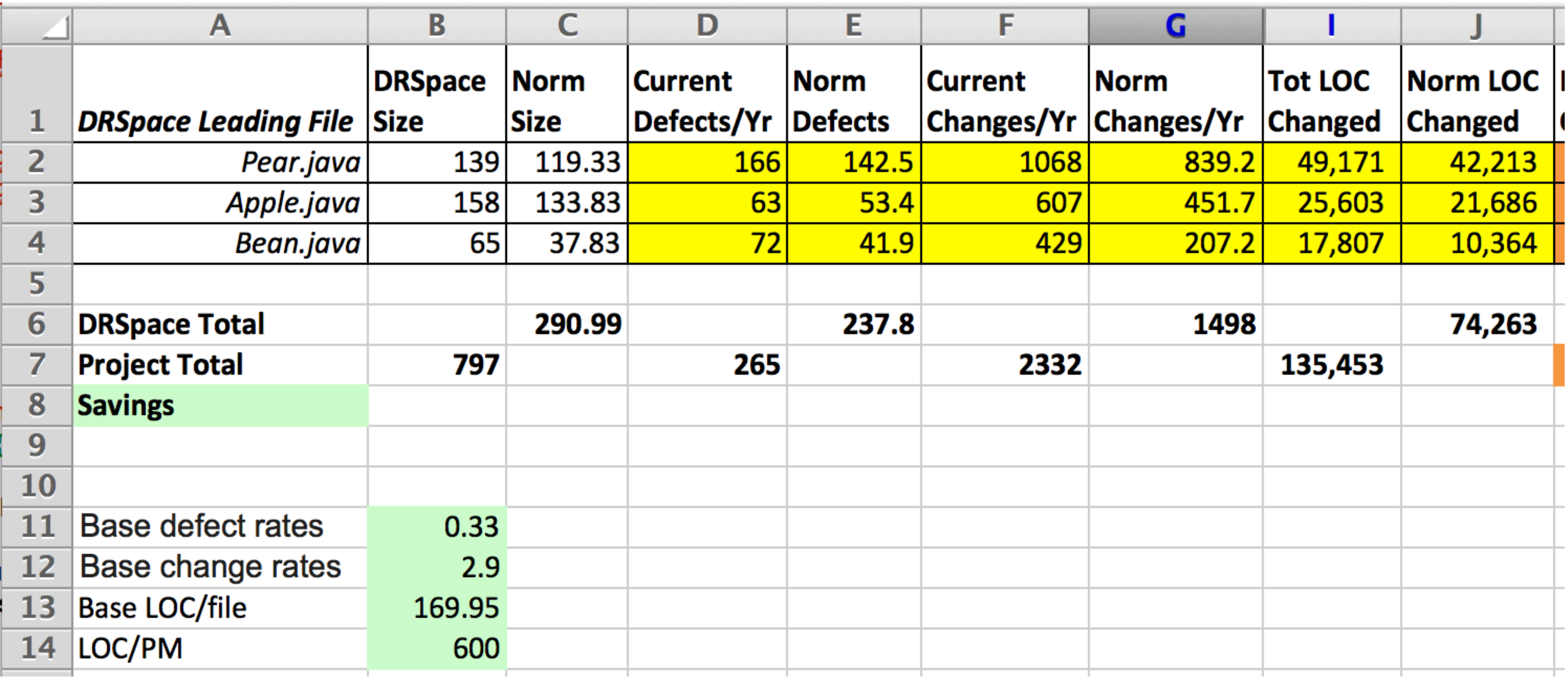
Consider the following analysis of the SoftServe project:

ECONOMIC ANALYSIS



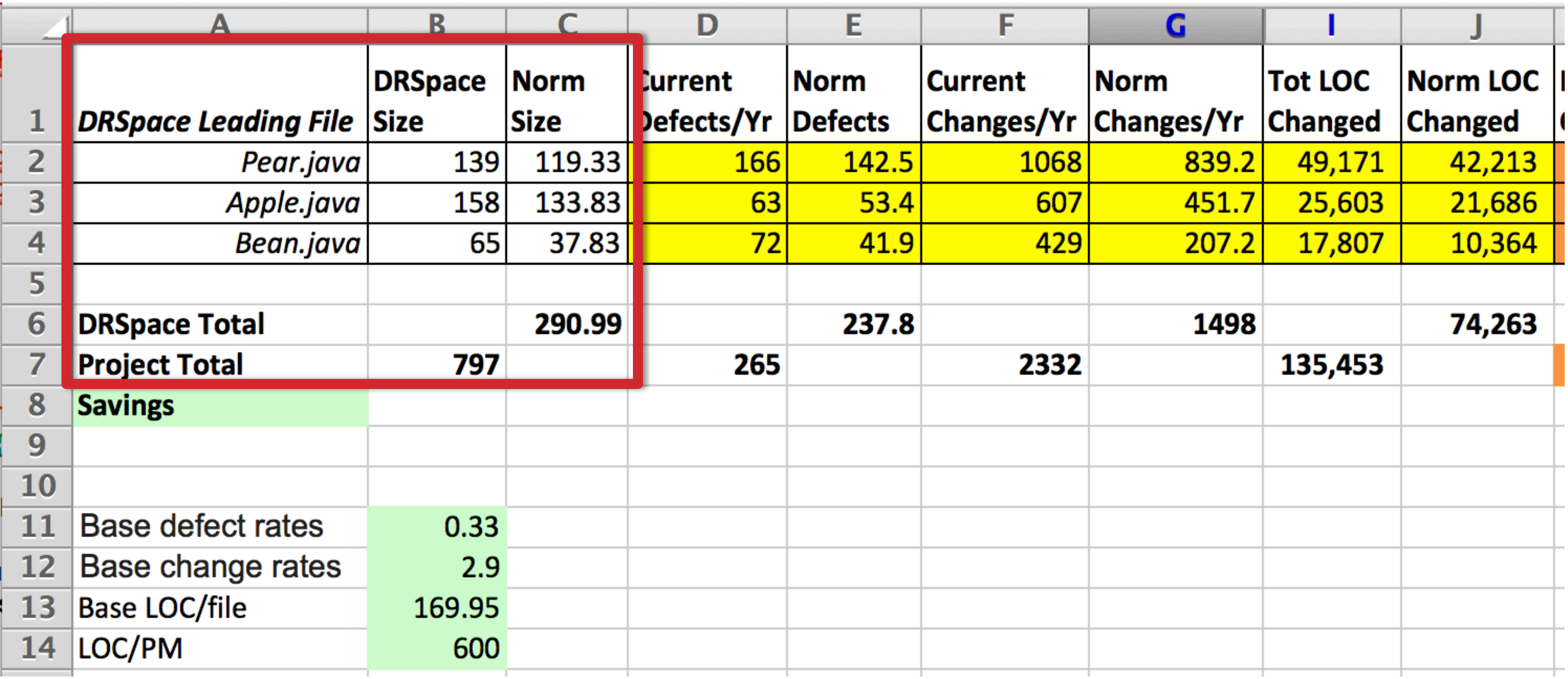
	A	B	C	D	E	F	G	I	J	K	L	M	N
1	***DRSpace Leading File***	**DRSpace Size**	**Norm Size**	**Current Defects/Yr**	**Norm Defects**	**Current Changes/Yr**	**Norm Changes/Yr**	**Tot LOC Changed**	**Norm LOC Changed**	**Refactor Cost (PM)**	**Norm Exp Defects/Yr**	**Norm Exp Changes/Yr**	**Norm Exp LOC Changed**
2	*Pear.java*	139	119.33	166	142.5	1068	839.2	49,171	42,213	5.5	39	346	20,281
3	*Apple.java*	158	133.83	63	53.4	607	451.7	25,603	21,686	7	44	388	22,745
4	*Bean.java*	65	37.83	72	41.9	429	207.2	17,807	10,364	1.5	12	110	6,429
5													
6	**DRSpace Total**		**290.99**		**237.8**		**1498**		**74,263**		**96.0**	**843.871**	**49,455**
7	**Project Total**	**797**		**265**		**2332**		**135,453**		**14**			
8	**Savings**										**142**	**654**	**24,808**
9													
10													
11	Base defect rates	0.33											
12	Base change rates	2.9										**Exp PM saved**	**41.35**
13	Base LOC/file	169.95											
14	LOC/PM	600											

ECONOMIC ANALYSIS



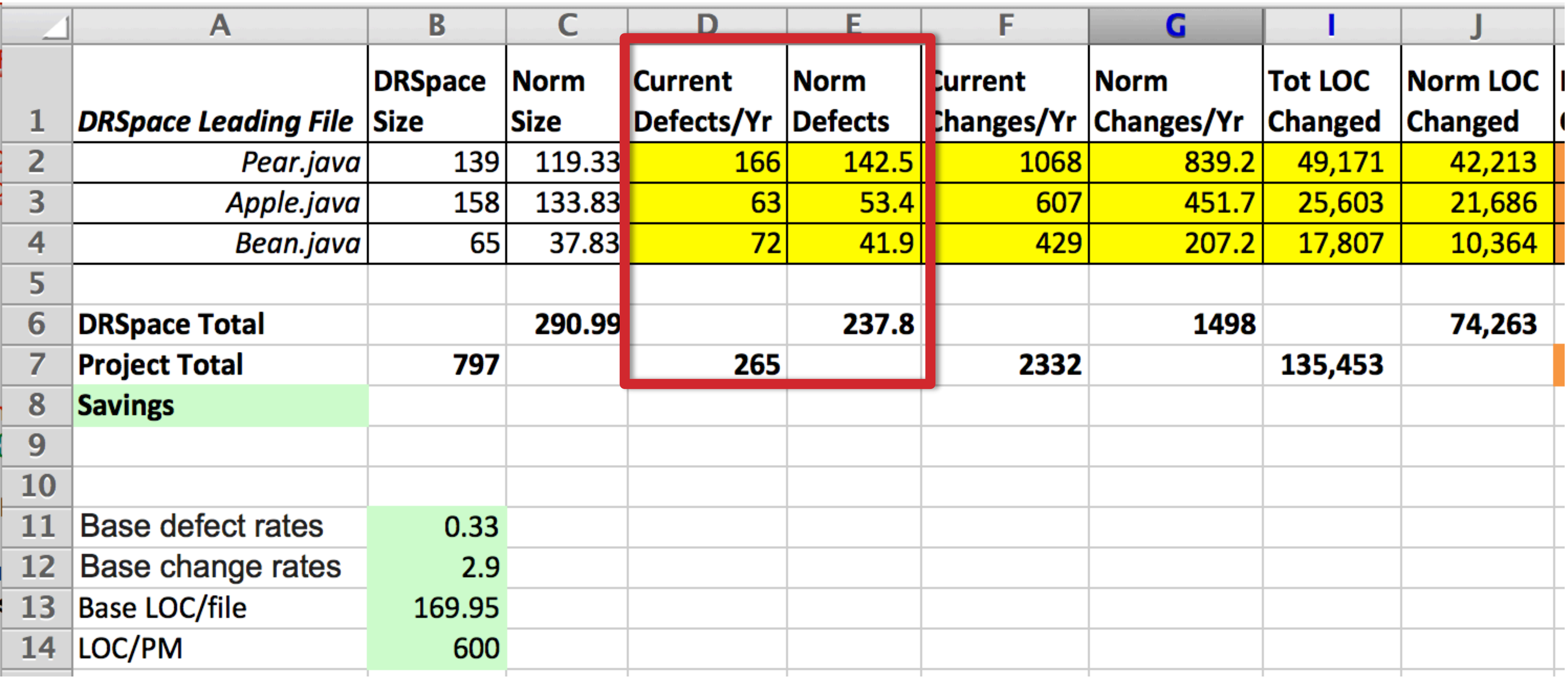
	A	B	C	D	E	F	G	I	J
1	***DRSpace Leading File***	**DRSpace Size**	**Norm Size**	**Current Defects/Yr**	**Norm Defects**	**Current Changes/Yr**	**Norm Changes/Yr**	**Tot LOC Changed**	**Norm LOC Changed**
2	*Pear.java*	139	119.33	166	142.5	1068	839.2	49,171	42,213
3	*Apple.java*	158	133.83	63	53.4	607	451.7	25,603	21,686
4	*Bean.java*	65	37.83	72	41.9	429	207.2	17,807	10,364
5									
6	**DRSpace Total**		**290.99**		**237.8**		**1498**		**74,263**
7	**Project Total**	**797**		**265**		**2332**		**135,453**	
8	**Savings**								
9									
10									
11	Base defect rates	0.33							
12	Base change rates	2.9							
13	Base LOC/file	169.95							
14	LOC/PM	600							

ECONOMIC ANALYSIS



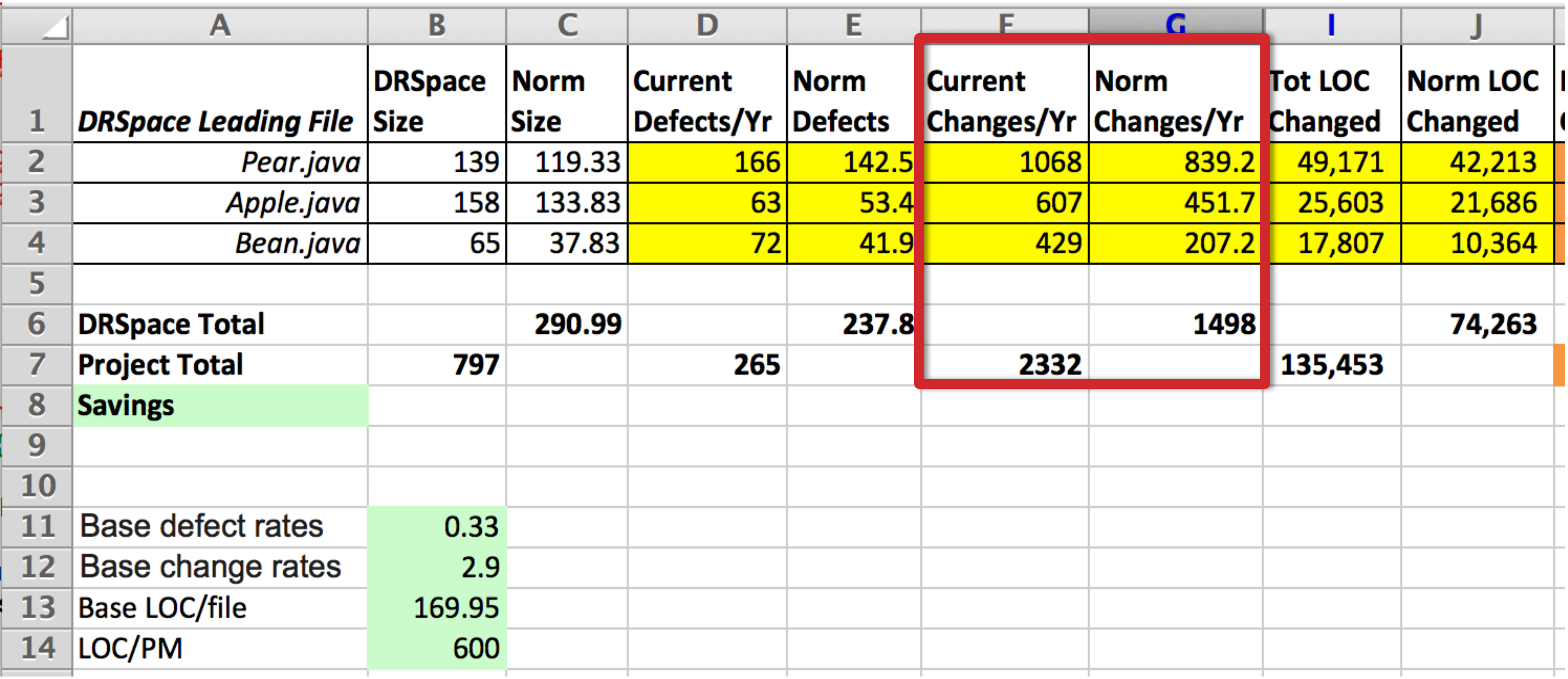
	A	B	C	D	E	F	G	I	J
1	***DRSpace Leading File***	**DRSpace Size**	**Norm Size**	**Current Defects/Yr**	**Norm Defects**	**Current Changes/Yr**	**Norm Changes/Yr**	**Tot LOC Changed**	**Norm LOC Changed**
2	*Pear.java*	139	119.33	166	142.5	1068	839.2	49,171	42,213
3	*Apple.java*	158	133.83	63	53.4	607	451.7	25,603	21,686
4	*Bean.java*	65	37.83	72	41.9	429	207.2	17,807	10,364
5									
6	**DRSpace Total**		**290.99**		**237.8**		**1498**		**74,263**
7	**Project Total**	**797**		**265**		**2332**		**135,453**	
8	**Savings**								
9									
10									
11	Base defect rates	0.33							
12	Base change rates	2.9							
13	Base LOC/file	169.95							
14	LOC/PM	600							

ECONOMIC ANALYSIS



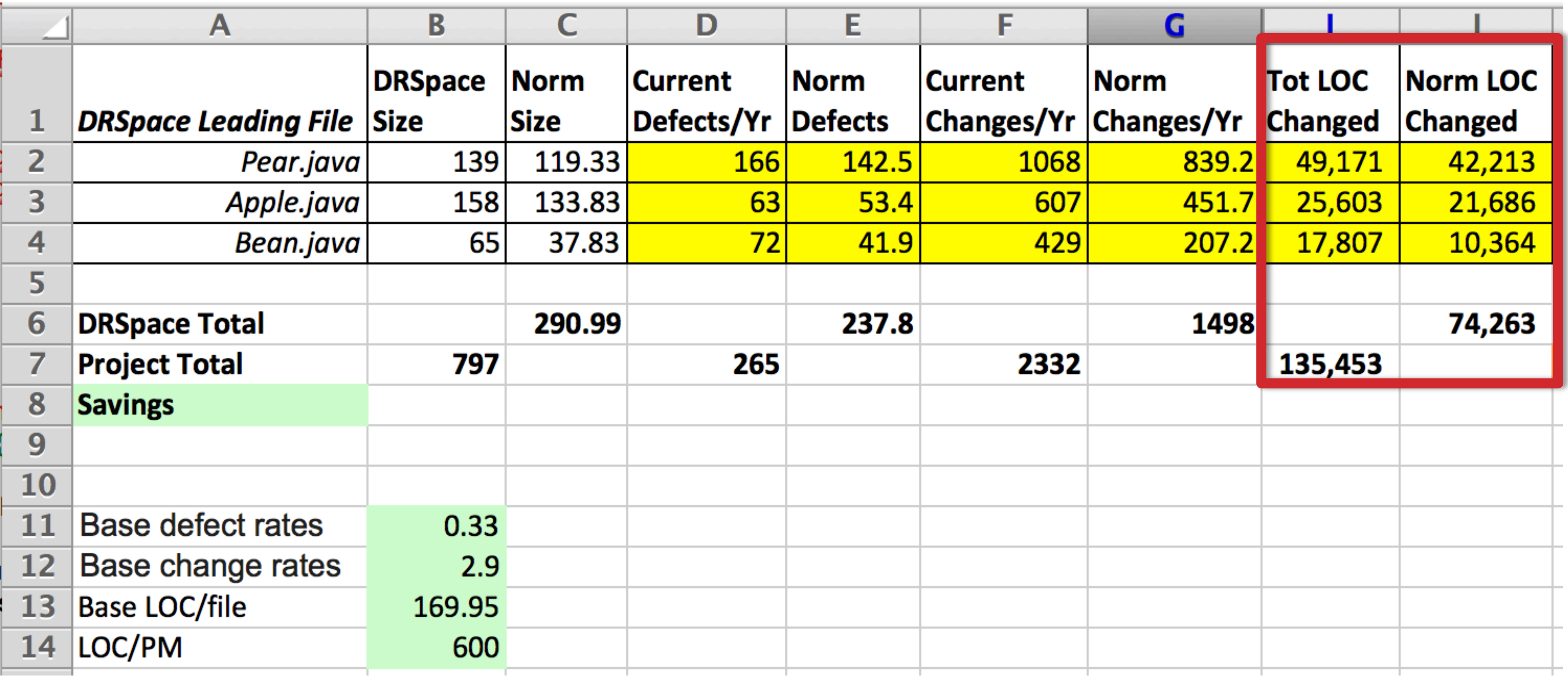
	A	B	C	D	E	F	G	I	J
1	***DRSpace Leading File***	**DRSpace Size**	**Norm Size**	**Current Defects/Yr**	**Norm Defects**	**Current Changes/Yr**	**Norm Changes/Yr**	**Tot LOC Changed**	**Norm LOC Changed**
2	*Pear.java*	139	119.33	166	142.5	1068	839.2	49,171	42,213
3	*Apple.java*	158	133.83	63	53.4	607	451.7	25,603	21,686
4	*Bean.java*	65	37.83	72	41.9	429	207.2	17,807	10,364
5									
6	**DRSpace Total**		**290.99**		**237.8**		**1498**		**74,263**
7	**Project Total**	**797**		**265**		**2332**		**135,453**	
8	**Savings**								
9									
10									
11	Base defect rates	0.33							
12	Base change rates	2.9							
13	Base LOC/file	169.95							
14	LOC/PM	600							

ECONOMIC ANALYSIS



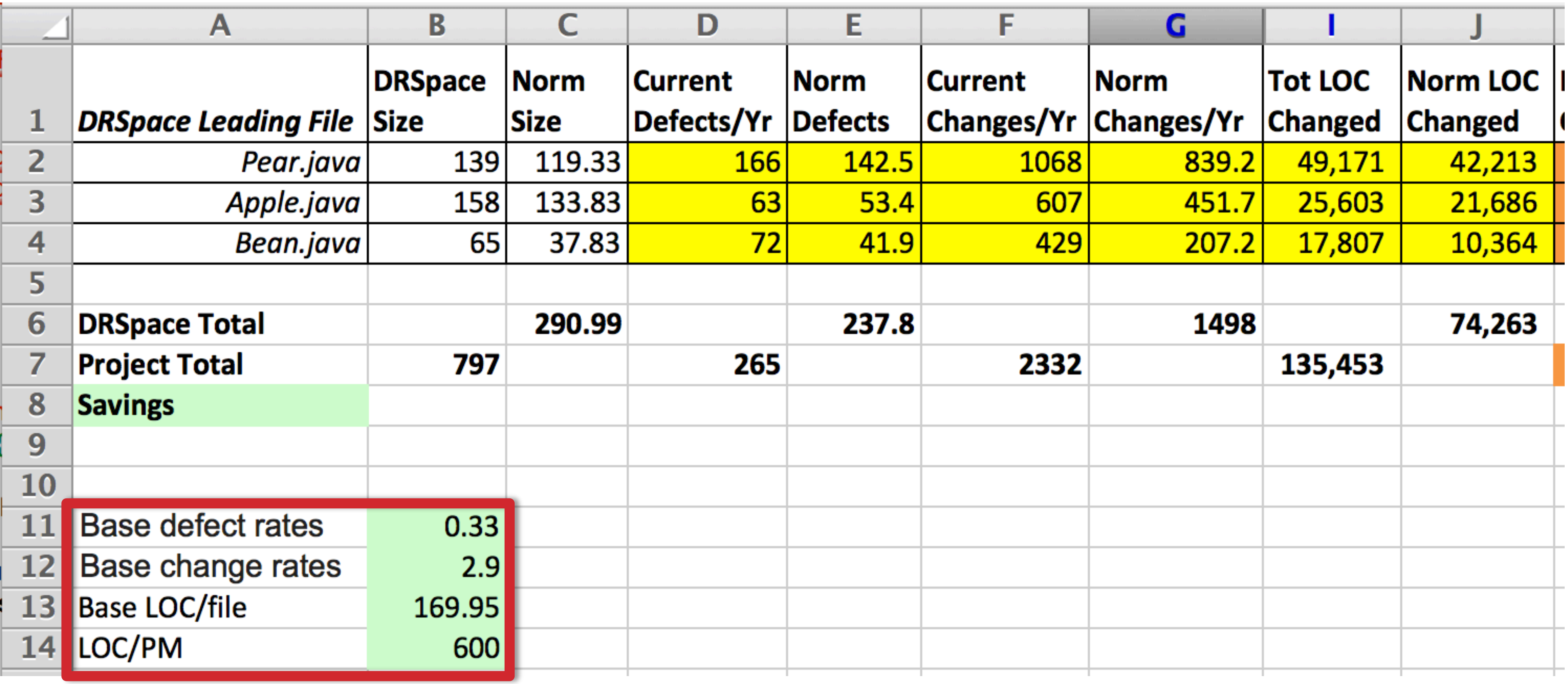
	A	B	C	D	E	F	G	I	J
1	***DRSpace Leading File***	**DRSpace Size**	**Norm Size**	**Current Defects/Yr**	**Norm Defects**	**Current Changes/Yr**	**Norm Changes/Yr**	**Tot LOC Changed**	**Norm LOC Changed**
2	*Pear.java*	139	119.33	166	142.5	1068	839.2	49,171	42,213
3	*Apple.java*	158	133.83	63	53.4	607	451.7	25,603	21,686
4	*Bean.java*	65	37.83	72	41.9	429	207.2	17,807	10,364
5									
6	**DRSpace Total**		**290.99**		**237.8**		**1498**		**74,263**
7	**Project Total**	**797**		**265**		**2332**		**135,453**	
8	**Savings**								
9									
10									
11	Base defect rates	0.33							
12	Base change rates	2.9							
13	Base LOC/file	169.95							
14	LOC/PM	600							

ECONOMIC ANALYSIS



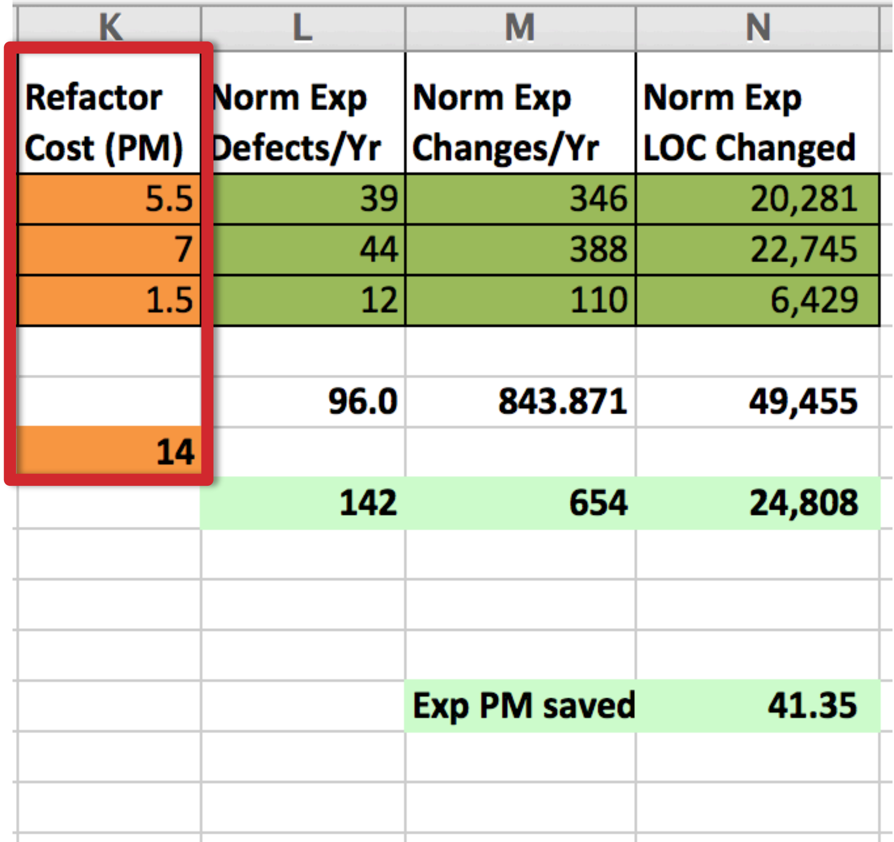
	A	B	C	D	E	F	G	I	J
1	***DRSpace Leading File***	**DRSpace Size**	**Norm Size**	**Current Defects/Yr**	**Norm Defects**	**Current Changes/Yr**	**Norm Changes/Yr**	**Tot LOC Changed**	**Norm LOC Changed**
2	*Pear.java*	139	119.33	166	142.5	1068	839.2	49,171	42,213
3	*Apple.java*	158	133.83	63	53.4	607	451.7	25,603	21,686
4	*Bean.java*	65	37.83	72	41.9	429	207.2	17,807	10,364
5									
6	**DRSpace Total**		**290.99**		**237.8**		**1498**		**74,263**
7	**Project Total**	**797**		**265**		**2332**		**135,453**	
8	**Savings**								
9									
10									
11	Base defect rates	0.33							
12	Base change rates	2.9							
13	Base LOC/file	169.95							
14	LOC/PM	600							

ECONOMIC ANALYSIS



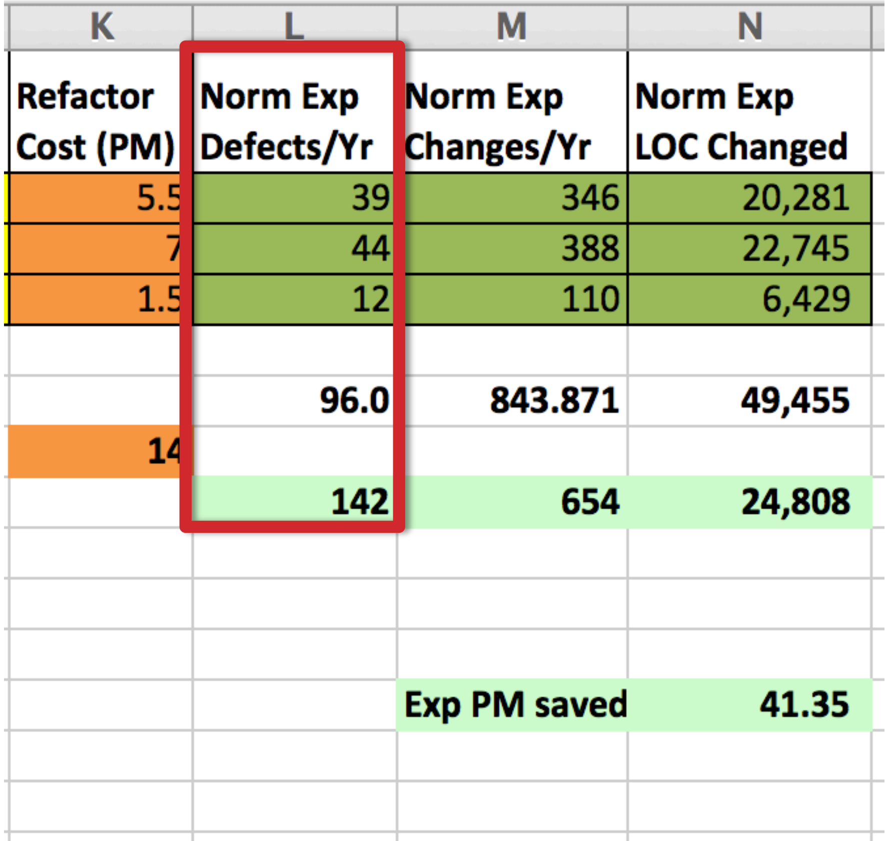
	A	B	C	D	E	F	G	I	J
1	***DRSpace Leading File***	**DRSpace Size**	**Norm Size**	**Current Defects/Yr**	**Norm Defects**	**Current Changes/Yr**	**Norm Changes/Yr**	**Tot LOC Changed**	**Norm LOC Changed**
2	*Pear.java*	139	119.33	166	142.5	1068	839.2	49,171	42,213
3	*Apple.java*	158	133.83	63	53.4	607	451.7	25,603	21,686
4	*Bean.java*	65	37.83	72	41.9	429	207.2	17,807	10,364
5									
6	**DRSpace Total**		**290.99**		**237.8**		**1498**		**74,263**
7	**Project Total**	**797**		**265**		**2332**		**135,453**	
8	**Savings**								
9									
10									
11	Base defect rates	0.33							
12	Base change rates	2.9							
13	Base LOC/file	169.95							
14	LOC/PM	600							

ECONOMIC ANALYSIS



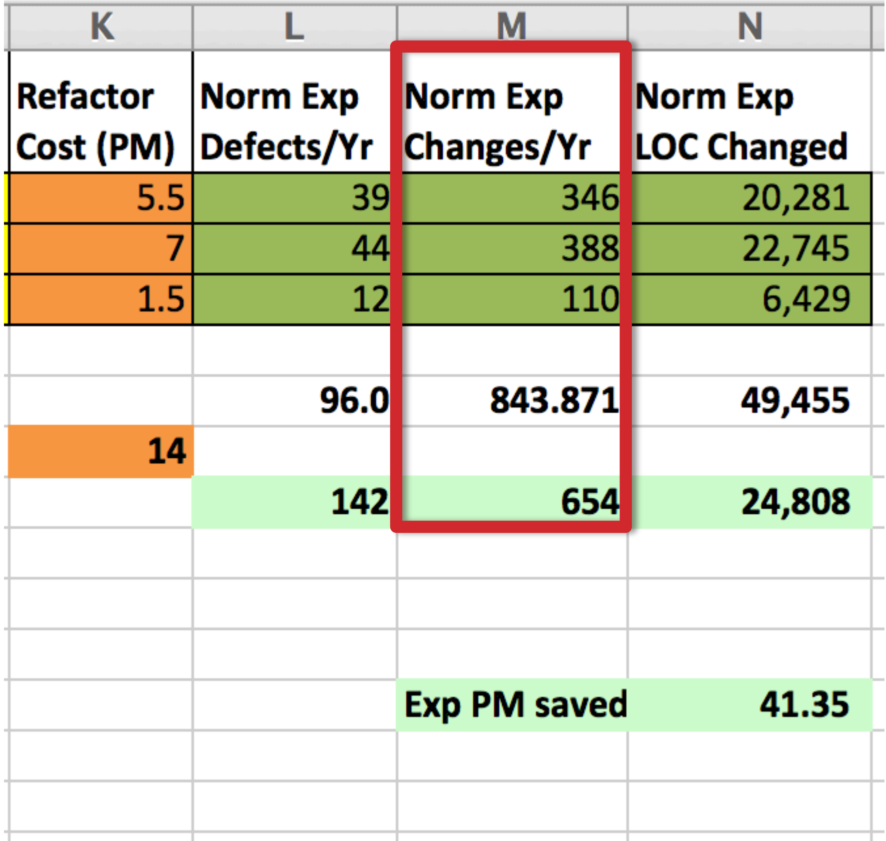
K	L	M	N
Refactor Cost (PM)	Norm Exp Defects/Yr	Norm Exp Changes/Yr	Norm Exp LOC Changed
5.5	39	346	20,281
7	44	388	22,745
1.5	12	110	6,429
	96.0	843.871	49,455
14			
	142	654	24,808
		Exp PM saved	41.35

ECONOMIC ANALYSIS



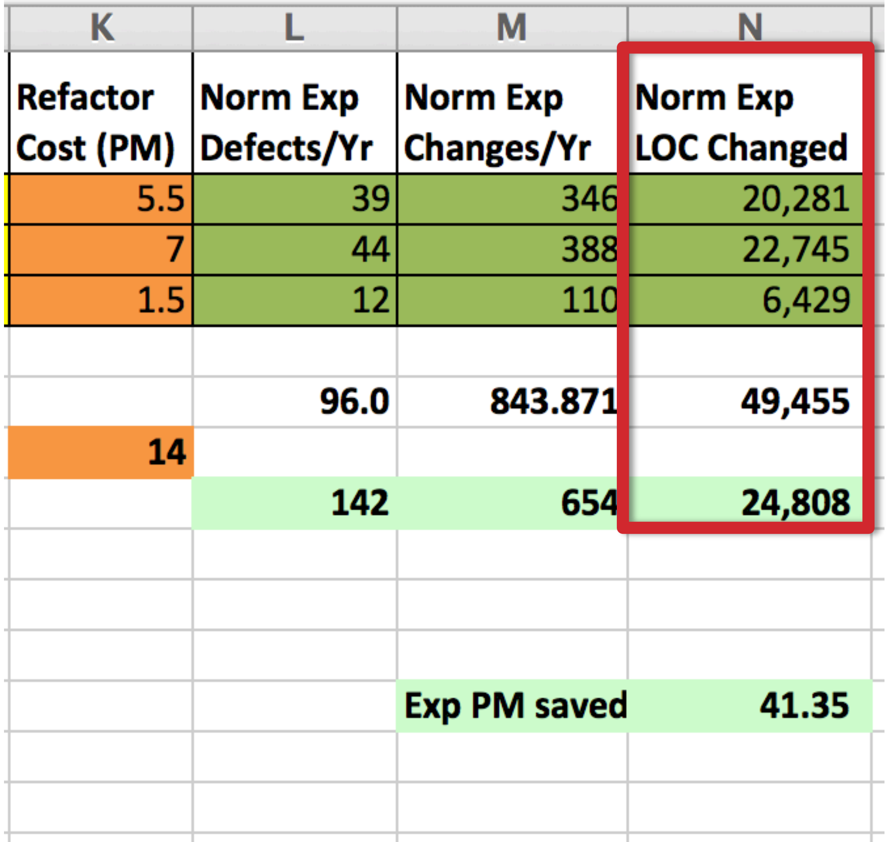
K	L	M	N
Refactor Cost (PM)	Norm Exp Defects/Yr	Norm Exp Changes/Yr	Norm Exp LOC Changed
5.5	39	346	20,281
7	44	388	22,745
1.5	12	110	6,429
	96.0	**843.871**	**49,455**
14			
	142	**654**	**24,808**
		Exp PM saved	**41.35**

ECONOMIC ANALYSIS



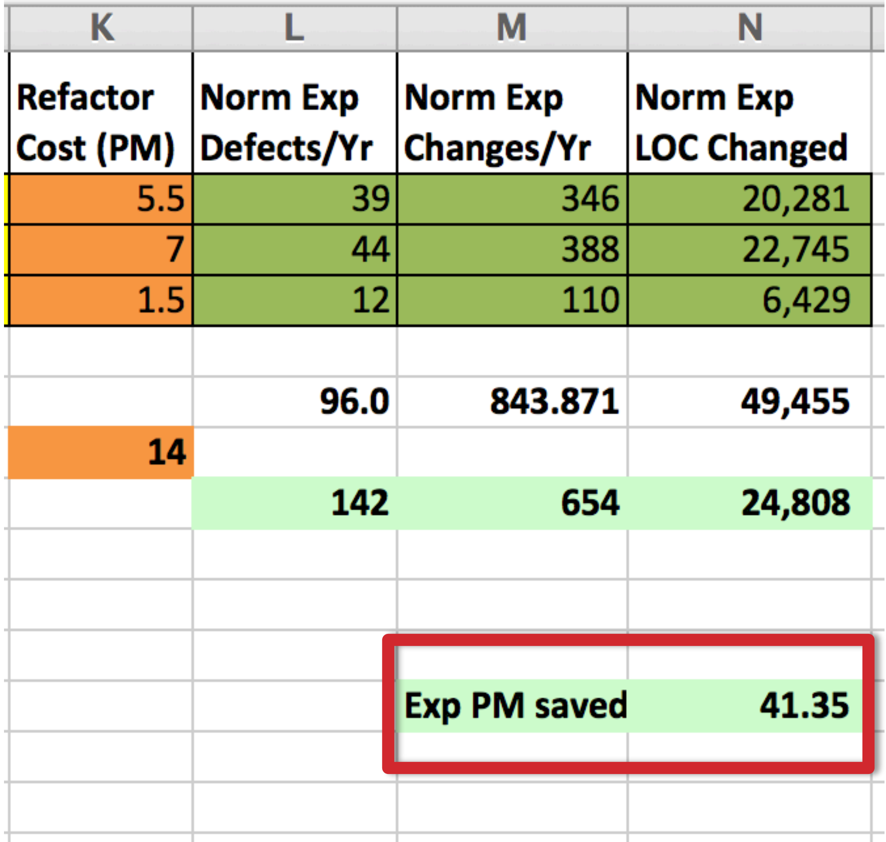
K	L	M	N
Refactor Cost (PM)	Norm Exp Defects/Yr	Norm Exp Changes/Yr	Norm Exp LOC Changed
5.5	39	346	20,281
7	44	388	22,745
1.5	12	110	6,429
	96.0	843.871	49,455
14			
	142	654	24,808
		Exp PM saved	41.35

ECONOMIC ANALYSIS



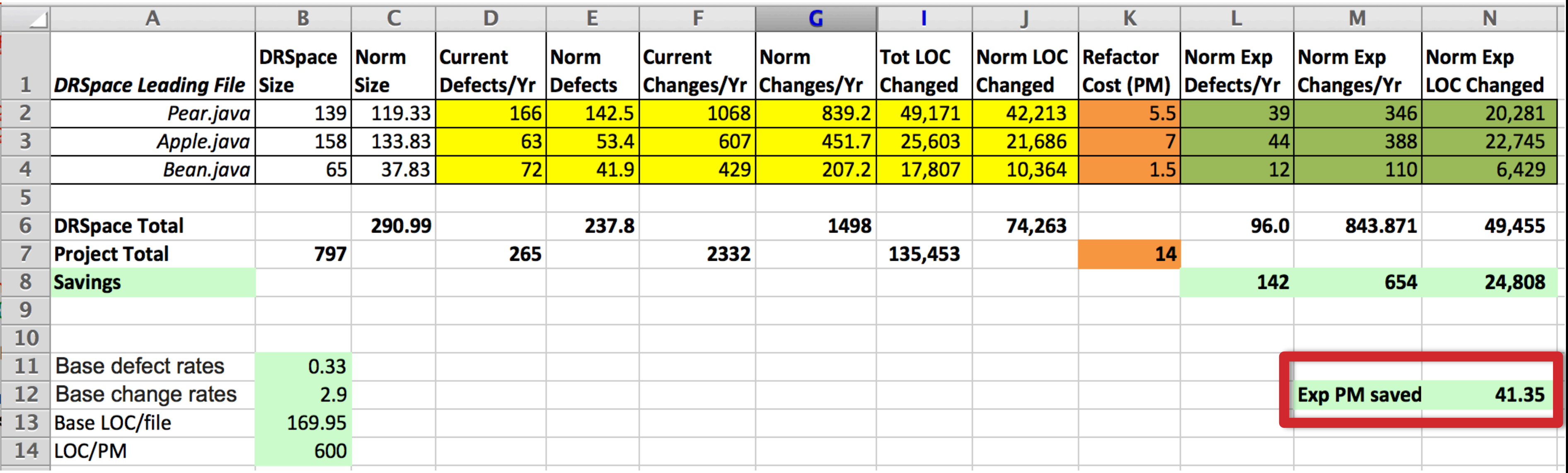
K	L	M	N
Refactor Cost (PM)	Norm Exp Defects/Yr	Norm Exp Changes/Yr	Norm Exp LOC Changed
5.5	39	346	20,281
7	44	388	22,745
1.5	12	110	6,429
	96.0	843.871	49,455
14			
	142	654	24,808
		Exp PM saved	41.35

ECONOMIC ANALYSIS



K	L	M	N
Refactor Cost (PM)	Norm Exp Defects/Yr	Norm Exp Changes/Yr	Norm Exp LOC Changed
5.5	39	346	20,281
7	44	388	22,745
1.5	12	110	6,429
	96.0	843.871	49,455
14			
	142	654	24,808
		Exp PM saved	41.35

ECONOMIC ANALYSIS



	A	B	C	D	E	F	G	I	J	K	L	M	N
1	***DRSpace Leading File***	**DRSpace Size**	**Norm Size**	**Current Defects/Yr**	**Norm Defects**	**Current Changes/Yr**	**Norm Changes/Yr**	**Tot LOC Changed**	**Norm LOC Changed**	**Refactor Cost (PM)**	**Norm Exp Defects/Yr**	**Norm Exp Changes/Yr**	**Norm Exp LOC Changed**
2	*Pear.java*	139	119.33	166	142.5	1068	839.2	49,171	42,213	5.5	39	346	20,281
3	*Apple.java*	158	133.83	63	53.4	607	451.7	25,603	21,686	7	44	388	22,745
4	*Bean.java*	65	37.83	72	41.9	429	207.2	17,807	10,364	1.5	12	110	6,429
5													
6	**DRSpace Total**		**290.99**		**237.8**		**1498**		**74,263**		**96.0**	**843.871**	**49,455**
7	**Project Total**	**797**		**265**		**2332**		**135,453**		**14**			
8	**Savings**										**142**	**654**	**24,808**
9													
10													
11	Base defect rates	0.33											
12	Base change rates	2.9										**Exp PM saved**	**41.35**
13	Base LOC/file	169.95											
14	LOC/PM	600											

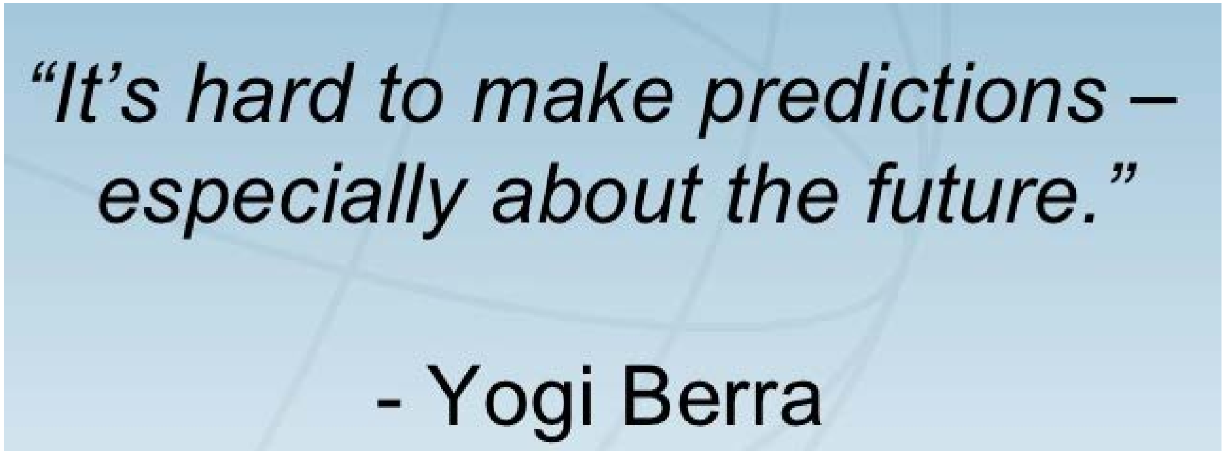
Result: ~300% ROI in the first year alone!

FOLLOW-ON

SoftServe is now refactoring the project, fixing the identified flaws.

This refactoring is expected to be complete this month.

Further data collection and hypothesis validation is planned.



“It’s hard to make predictions – especially about the future.”

- Yogi Berra

TAKE-AWAYS

Architectural flaws lead to quality issues.

We can locate these flaws!

We can not fix the quality issues without fixing the underlying flaws!

TAKE-AWAYS

This analysis allows us to plan refactoring strategies and make informed, *economics-based* decisions about if and how to refactor.

This work was supported in part by the U.S. National Science Foundation under grants CCF-0916891, CCF-1065189, CCF-1116980 and DUE-0837665.