



Release Your Refactoring Superpower

By:

Adam Culp

Twitter: [@adamculp](https://twitter.com/adamculp)



Release Your Refactoring Superpower

- **About me**

- OSS Contributor
- PHP Certified
- Zend Certification Advisory Board
- PHP-Fig voting member (IBM i Toolkit)
- Consultant at Zend Technologies
- Organizer SoFloPHP (South Florida)
- Organizer SunshinePHP (Miami)
- Long distance (ultra) runner
- Photography Enthusiast
 - Judo Black Belt Instructor



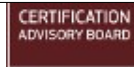
Release Your Refactoring Superpower

- About me

- OSS Contributor
- PHP Certified
- Zend Certification Advisory Board
- PHP-Fig voting member (IPM + Toolkit)
- Consultant at Zen
- Organizer SoFloPHP
- Organizer Sunshine
- Long distance (ultra)
- Photography Enth
- Judo Black Belt In



**I am the
PHP Ninja!!!**



Release Your Refactoring Superpower

- **Fan of iteration**

- Pretty much everything requires iteration to do well:
 - Long distance running
 - Judo
 - Development
 - Evading project managers
 - Refactoring!



Release Your Refactoring Superpower

- **What Can I Do?**
 - Estimation

Release Your Refactoring Superpower

- **What Can I Do?**
 - Estimation
 - Coding (actual refactoring)

Release Your Refactoring Superpower

- **What Can I Do?**
 - Estimation
 - Coding (actual refactoring)
 - Algorithms

Release Your Refactoring Superpower

- **What Can I Do?**
 - Estimation
 - Coding (actual refactoring)
 - Algorithms
 - Convince Business

Release Your Refactoring Superpower

- **What Can I Do?**
 - Estimation
 - Coding (actual refactoring)
 - Algorithms
 - Convince Business
 - Silver Bullet



Release Your Refactoring Superpower

- **Modernization?**
 - How?
 - New infrastructure (servers, technology, etc.)

Release Your Refactoring Superpower

- **Modernization?**
 - How?
 - New infrastructure (servers, technology, etc.)
 - New frameworks or libraries

Release Your Refactoring Superpower

- **Modernization?**
 - How?
 - New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language

Release Your Refactoring Superpower

- **Modernization?**
 - How?
 - New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language
 - New DB

Release Your Refactoring Superpower

- **Modernization?**
 - How?
 - New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language
 - New DB
 - Why?
 - Desire

Release Your Refactoring Superpower

- **Modernization?**
 - How?
 - New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language
 - New DB
 - Why?
 - Desire
 - Bored

Release Your Refactoring Superpower

- **Modernization?**
 - How?
 - New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language
 - New DB
 - Why?
 - Desire
 - Bored
 - Perceived need

Release Your Refactoring Superpower

- **Modernization?**
 - How?
 - New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language
 - New DB
 - Why?
 - Desire
 - Bored
 - Perceived need
 - To gain something
 - Speed
 - Functionality

Release Your Refactoring Superpower

- **Modernization?**
 - **How?**
 - New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language
 - New DB
 - **Why?**
 - Desire
 - Bored
 - Perceived need
 - To gain something
 - Speed
 - Functionality
 - **When?**
 - Next 6 months, year(s), decade

Release Your Refactoring Superpower

- **Modernization?**

- **How?**

- New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language
 - New DB

- **Why?**

- Desire
 - Bored
 - Perceived need
 - To gain something
 - Speed
 - Functionality

- **When?**

- Next 6 months, year(s), decade
 - Realistic time

Release Your Refactoring Superpower

- **Modernization?**

- **How?**

- New infrastructure (servers, technology, etc.)
 - New frameworks or libraries
 - New programming language
 - New DB

- **Why?**

- Desire
 - Bored
 - Perceived need
 - To gain something
 - Speed
 - Functionality

- **When?**

- Next 6 months, year(s), decade
 - Realistic time
 - **NOW!**



Release Your Refactoring Superpower

- Rewrite FTW!



Release Your Refactoring Superpower

- **Typical Loop**
 - Business Responses
 - No time



Release Your Refactoring Superpower

- **Typical Loop**

- Business Responses

- No time
 - No money



Release Your Refactoring Superpower

- **Typical Loop**

- **Business Responses**

- No time
 - No money
 - No need



Release Your Refactoring Superpower

- **Typical Loop**

- **Business Responses**

- No time
 - No money
 - No need
 - Things are “good enough”



Release Your Refactoring Superpower

- The Fix



Release Your Refactoring Superpower

- **Case Study**

- **Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company - 2017 (by James Crotty, Ivan Horrocks) -**

<https://www.sciencedirect.com/science/article/pii/S2210832716301260#b0025>

- 2001 Brooke and Ramage defined legacy as:
 - Old information system remaining in operation within an Organization

Release Your Refactoring Superpower

- **Case Study**

- **Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company - 2017 (by James Crotty, Ivan Horrocks) -**

<https://www.sciencedirect.com/science/article/pii/S2210832716301260#b0025>

- **2001 Brooke and Ramage defined legacy as:**
 - Old information system remaining in operation within an Organization
 - Business critical, resisting modification as failure would cause significant impact on business

Release Your Refactoring Superpower

- **Case Study**

- **Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company - 2017 (by James Crotty, Ivan Horrocks) -**

<https://www.sciencedirect.com/science/article/pii/S2210832716301260#b0025>

- **2001 Brooke and Ramage defined legacy as:**
 - Old information system remaining in operation within an Organization
 - Business critical, resisting modification as failure would cause significant impact on business
 - Based on outdated technology but critical day-to-day operations

Release Your Refactoring Superpower

- **Case Study**

- **Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company - 2017 (by James Crotty, Ivan Horrocks) -**

<https://www.sciencedirect.com/science/article/pii/S2210832716301260#b0025>

- **2001 Brooke and Ramage defined legacy as:**
 - Old information system remaining in operation within an Organization
 - Business critical, resisting modification as failure would cause significant impact on business
 - Based on outdated technology but critical day-to-day operations
 - Built when processing and storage was much more expensive

Release Your Refactoring Superpower

- **Case Study**

- **Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company - 2017 (by James Crotty, Ivan Horrocks) -**

<https://www.sciencedirect.com/science/article/pii/S2210832716301260#b0025>

- **2001 Brooke and Ramage defined legacy as:**
 - Old information system remaining in operation within an Organization
 - Business critical, resisting modification as failure would cause significant impact on business
 - Based on outdated technology but critical day-to-day operations
 - Built when processing and storage was much more expensive
 - Poorly documented

Release Your Refactoring Superpower

- **Case Study**

- **Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company - 2017 (by James Crotty, Ivan Horrocks) -**

<https://www.sciencedirect.com/science/article/pii/S2210832716301260#b0025>

- **2001 Brooke and Ramage defined legacy as:**
 - Old information system remaining in operation within an Organization
 - Business critical, resisting modification as failure would cause significant impact on business
 - Based on outdated technology but critical day-to-day operations
 - Built when processing and storage was much more expensive
 - Poorly documented
 - Lack of design

Release Your Refactoring Superpower

- **Case Study**
 - Modernization Drivers
 - Skillset shortages (old technologies)

Release Your Refactoring Superpower

- **Case Study**
 - **Modernization Drivers**
 - Skillset shortages (old technologies)
 - Technical needs

Release Your Refactoring Superpower

- **Case Study**
 - **Modernization Drivers**
 - Skillset shortages (old technologies)
 - Technical needs
 - Business needs

Release Your Refactoring Superpower

- **Case Study**
 - **Modernization Drivers**
 - Skillset shortages (old technologies)
 - Technical needs
 - Business needs
 - Personal bias

Release Your Refactoring Superpower

- **Case Study**
 - Cost Reduction Strategies
 - Ordinary maintenance

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering
 - Restructuring

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering
 - Restructuring
 - Re-engineering

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering
 - Restructuring
 - Re-engineering
 - Migration

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering
 - Restructuring
 - Re-engineering
 - Migration
 - Discard

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering
 - Restructuring
 - Re-engineering
 - Migration
 - Discard
 - Wrapping

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering
 - Restructuring
 - Re-engineering
 - Migration
 - Discard
 - Wrapping
 - Outsource?

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering
 - Restructuring
 - Re-engineering
 - Migration
 - Discard
 - Wrapping
 - Outsource?
 - Freeze

Release Your Refactoring Superpower

- **Case Study**
 - **Cost Reduction Strategies**
 - Ordinary maintenance
 - Reverse engineering
 - Restructuring
 - Re-engineering
 - Migration
 - Discard
 - Wrapping
 - Outsource?
 - Freeze
 - Carry On

Release Your Refactoring Superpower

- **Case Study**

- **Cost Reduction Strategies**

- Ordinary maintenance
 - Reverse engineering
 - Restructuring
 - Re-engineering
 - Migration
 - Discard
 - Wrapping
 - Outsource?
 - Freeze
 - Carry On
 - Replacement with commercial off-the-shelf software and discarding

Release Your Refactoring Superpower

- **How Do We Know?**

- **Measurement**

- A Method for Assessing Legacy Systems for Evolution - 1998 (by Jane Ransom, Ian Sommerville, and Ian Warren) -

- <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.9889&rep=rep1&type=pdf>

- Legacy = business critical = cost not justifiable

Release Your Refactoring Superpower

- **How Do We Know?**

- **Measurement**

- A Method for Assessing Legacy Systems for Evolution - 1998 (by Jane Ransom, Ian Sommerville, and Ian Warren) -
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.9889&rep=rep1&type=pdf>
 - Legacy = business critical = cost not justifiable
 - Company and project specific

Release Your Refactoring Superpower

- **How Do We Know?**

- **Measurement**

- **A Method for Assessing Legacy Systems for Evolution - 1998 (by Jane Ransom, Ian Sommerville, and Ian Warren) -**

- <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.9889&rep=rep1&type=pdf>

- Legacy = business critical = cost not justifiable
 - Company and project specific
 - Continuously refined

Release Your Refactoring Superpower

- **How Do We Know?**

- **Measurement**

- **A Method for Assessing Legacy Systems for Evolution - 1998 (by Jane Ransom, Ian Sommerville, and Ian Warren) -**

- <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.9889&rep=rep1&type=pdf>

- Legacy = business critical = cost not justifiable
 - Company and project specific
 - Continuously refined
 - Gains depth of understanding of business

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- Decision Model for Legacy Systems - 1999 (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>
 - Based more upon organizational points
 - Boundary: the unit of analysis

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- Decision Model for Legacy Systems - 1999 (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>

- Based more upon organizational points

- Boundary: the unit of analysis
 - Vision: global summary of the unit

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- Decision Model for Legacy Systems - 1999 (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>

- Based more upon organizational points

- Boundary: the unit of analysis
 - Vision: global summary of the unit
 - Logic: rationale for vision

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- **Decision Model for Legacy Systems - 1999** (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>

- **Based more upon organizational points**

- **Boundary:** the unit of analysis
 - **Vision:** global summary of the unit
 - **Logic:** rationale for vision
 - **Structure:** of the organisation

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- Decision Model for Legacy Systems - 1999 (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>

- Based more upon organizational points

- Boundary: the unit of analysis
 - Vision: global summary of the unit
 - Logic: rationale for vision
 - Structure: of the organisation
 - Roles: organizational roles of people

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- **Decision Model for Legacy Systems - 1999** (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>

- **Based more upon organizational points**

- **Boundary:** the unit of analysis
 - **Vision:** global summary of the unit
 - **Logic:** rationale for vision
 - **Structure:** of the organisation
 - **Roles:** organizational roles of people
 - **View of information:** resource analysis

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- **Decision Model for Legacy Systems - 1999** (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>

- **Based more upon organizational points**

- **Boundary:** the unit of analysis
 - **Vision:** global summary of the unit
 - **Logic:** rationale for vision
 - **Structure:** of the organisation
 - **Roles:** organizational roles of people
 - **View of information:** resource analysis
 - **Costs:** major costs, both financial and nonfinancial

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- **Decision Model for Legacy Systems - 1999** (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>

- **Based more upon organizational points**

- **Boundary:** the unit of analysis
 - **Vision:** global summary of the unit
 - **Logic:** rationale for vision
 - **Structure:** of the organisation
 - **Roles:** organizational roles of people
 - **View of information:** resource analysis
 - **Costs:** major costs, both financial and nonfinancial
 - **Benefits:** both financial and nonfinancial

Release Your Refactoring Superpower

- **How Do We Know?**

- **Criteria**

- **Decision Model for Legacy Systems - 1999** (by . H. Bennett, M. Ramage, and M. Munro) - <ftp://ftp.inf.puc-rio.br/pub/docs/FomularioSolicitacoes/mariliaGFerreira-09-13-7.pdf>

- **Based more upon organizational points**

- **Boundary:** the unit of analysis
 - **Vision:** global summary of the unit
 - **Logic:** rationale for vision
 - **Structure:** of the organisation
 - **Roles:** organizational roles of people
 - **View of information:** resource analysis
 - **Costs:** major costs, both financial and nonfinancial
 - **Benefits:** both financial and nonfinancial
 - **Risks:** major sources of risk

Release Your Refactoring Superpower

- **How Do We Know?**

- **Structure**

- A Framework to Assess Legacy Software Systems - 2014 (by Basem Y. Alkazemi) -

- <https://pdfs.semanticscholar.org/da50/7665a6c3bacb5559996bd436a9f76aa4e5a7.pdf>

- **Strategies**

- **Replacing**

TABLE II.
WEIGHTED DECISION-MAKING GRID

Pros	Score	Cons	Score
System functionality	5	Lack of web-services	3
Modification delivery time	5	Usability problems	3
Team support	4	Report generation	3
Technical Requirements	2	Architectural Style	2
DB technology	3	Oracle 6i problems	3
License	3	Business process problem	5
Overall	22		19

Release Your Refactoring Superpower

- **How Do We Know?**

- **Structure**

- A Framework to Assess Legacy Software Systems - 2014 (by Basem Y. Alkazemi) -

- <https://pdfs.semanticscholar.org/da50/7665a6c3bacb5559996bd436a9f76aa4e5a7.pdf>

- **Strategies**

- Replacing
 - Maintaining

TABLE II.
WEIGHTED DECISION-MAKING GRID

Pros	Score	Cons	Score
System functionality	5	Lack of web-services	3
Modification delivery time	5	Usability problems	3
Team support	4	Report generation	3
Technical Requirements	2	Architectural Style	2
DB technology	3	Oracle 6i problems	3
License	3	Business process problem	5
Overall	22		19

Release Your Refactoring Superpower

- **How Do We Know?**

- **Structure**

- A Framework to Assess Legacy Software Systems - 2014 (by Basem Y. Alkazemi) -

- <https://pdfs.semanticscholar.org/da50/7665a6c3bacb5559996bd436a9f76aa4e5a7.pdf>

- **Strategies**

- Replacing
 - Maintaining
 - Re-architecting

TABLE II.
WEIGHTED DECISION-MAKING GRID

Pros	Score	Cons	Score
System functionality	5	Lack of web-services	3
Modification delivery time	5	Usability problems	3
Team support	4	Report generation	3
Technical Requirements	2	Architectural Style	2
DB technology	3	Oracle 6i problems	3
License	3	Business process problem	5
Overall	22		19

Release Your Refactoring Superpower

- **How Do We Know?**

- **Structure**

- A Framework to Assess Legacy Software Systems - 2014 (by Basem Y. Alkazemi) -

- <https://pdfs.semanticscholar.org/da50/7665a6c3bacb5559996bd436a9f76aa4e5a7.pdf>

- **Strategies**

- Replacing
 - Maintaining
 - Re-architecting
 - Extending by wrapping

TABLE II.
WEIGHTED DECISION-MAKING GRID

Pros	Score	Cons	Score
System functionality	5	Lack of web-services	3
Modification delivery time	5	Usability problems	3
Team support	4	Report generation	3
Technical Requirements	2	Architectural Style	2
DB technology	3	Oracle 6i problems	3
License	3	Business process problem	5
Overall	22		19

Release Your Refactoring Superpower

- **How Do We Know?**

- Application

- A Decisional Framework to Measure System Dimensions of Legacy Application for Rejuvenation through Reengineering - 2011 (by Er. Anand Rajavat, Dr. (Mrs.) Vrinda Tokekar) -
<https://www.ijcaonline.org/volume16/number2/pxc3872674.pdf>

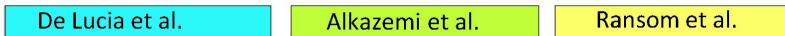
- System domain

- Customer requirements
 - Orgs strategic goals
 - Operational env
 - Risk management
 - i. Organizational
 - ii. Resource
 - iii. Development
 - iv. Personal
 - v. User Requirement
 - vi. Specialization
 - vii. Team
 - viii. Communication

Release Your Refactoring Superpower



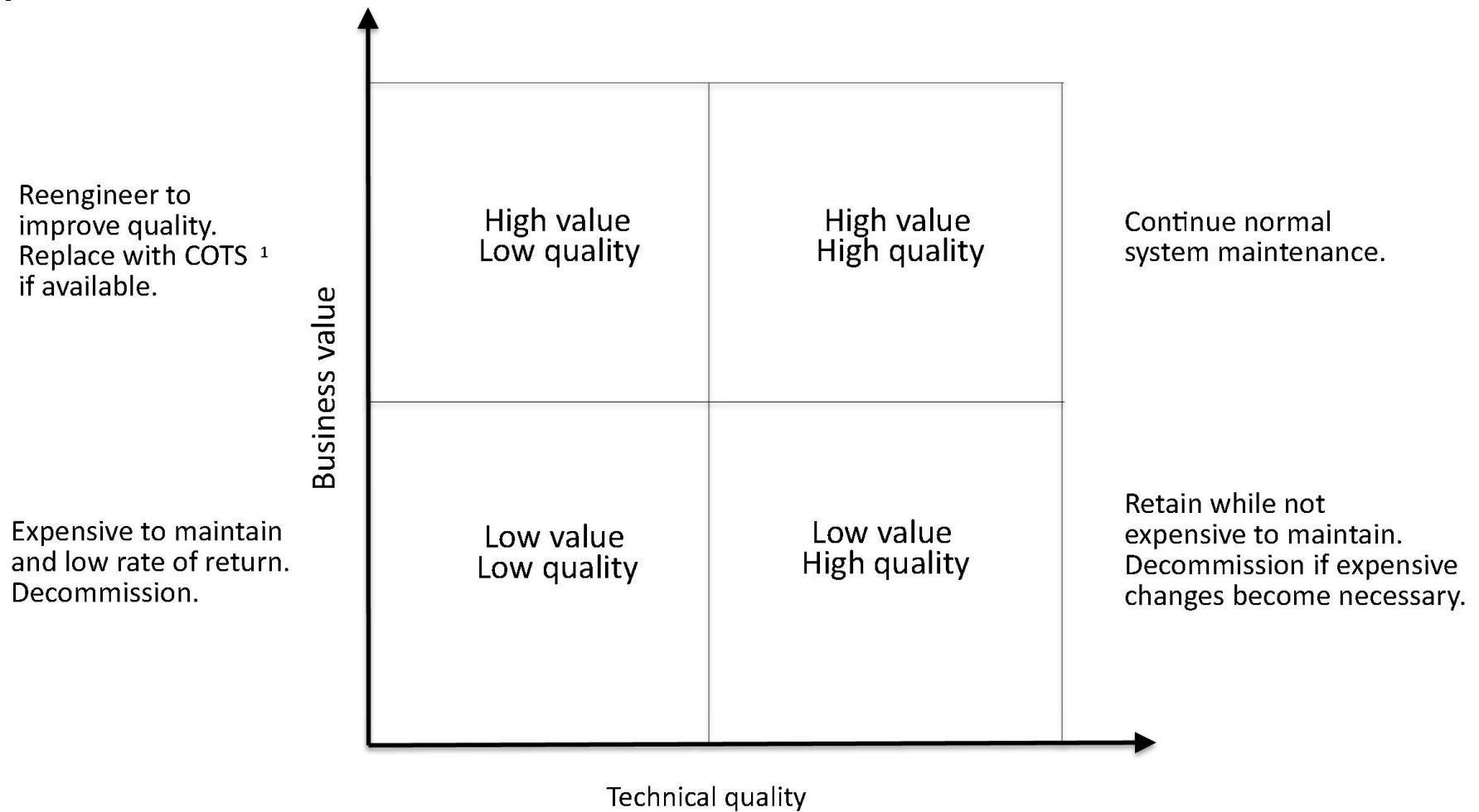
Legend



Source

Adapted from Alkazemi et al. [5], De Lucia et al. [7] and Ransom et al. [8]

Release Your Refactoring Superpower



Source: Sommerville [14]
¹Commercial off-the-shelf system

Release Your Refactoring Superpower

- **Case Study**

- **Example Assessment**

- **Step #1 - Does application meet or exceed definition of “Legacy”?**
 - **Answers: Yes, No, Maybe, Don’t know**
 - Business Critical
 - Old
 - Changed to meet organizational needs
 - Degrades as changes made
 - Maintenance cost increase as changes made
 - Obsolete languages
 - Poor, if any, documentation
 - Inadequate data management
 - Limited support capability
 - Limited support capacity
 - Lacks architecture to evolve to meet emerging requirements

Release Your Refactoring Superpower

- **Case Study**

- **Example Assessment**

- **Step #1 - Technical value attribute assessment**

- **Answers: Yes, No, Don't know**

- **Maintainability**

- **LOC**
 - **Control Flow**
 - **Cyclomatic complexity**
 - **Dead code fate**

- **Decompostability/Architecture**

- **Modularity**
 - **% of modules with separation of concerns**
 - **Consumption**
 - **Extensibility**
 - **Style**
 - **Interoperability**

Release Your Refactoring Superpower

- **Case Study**

- **Example Assessment**

- **Step #1 - Cont'd**

- **Answers: Yes, No, Don't know**

- **Deterioration**

- Backlog increase
 - Defect rate increase
 - Response-time increase
 - Maintenance time per request increase

- **Obsolescence**

- System age
 - Operating system version
 - Hardware version
 - Technical support availability
 - Security
 - Legality
 - System evolution required for business goals?

Release Your Refactoring Superpower

- **Case Study**

- **Example Assessment**

- **Step #2 - Business value attribute assessment**

- **Answers: Yes, No, Don't know**

- **Economic value**

- **Market value**
 - **Profitability index**
 - **IRR**

- **Data value**

- **% of mission critical archives**
 - **% of application dependent archives**

- **Utility**

- **Business function coverage rate**
 - **Actual usage frequency**
 - **Customer/user satisfaction metric**

- **Specialization**

- **% of highly specialized functions**
 - **% of generic functions**

Release Your Refactoring Superpower

- **Case Study**

- **Example Assessment**

- **Step #3 - Organizational infrastructure attribute assessment**
 - **Answers: Yes, No, Don't know**
 - Development & maintenance internal or outsourced?
 - Technical maturity
 - Commitment to training
 - Skill level of system support
 - Response to change

Release Your Refactoring Superpower

- **Case Study**

- **Example Assessment**

- **Step #4 - Calculations**

- Calculate all responses to 1 - 5 values (don't know = 0)
 - To easily plot on decisional matrix

Business value attributes	Y	N	D/K^a	1	2	3	4	5	0
<i>Economic value</i>									
Market value	8	1	1				1	9	4.90

Release Your Refactoring Superpower

- **Case Study**

- **Example Assessment**

- **Step #5 - Conversion**

- Convert Y, N, DK to numeric values and Avg %

$$\textit{individual business attribute value} = \frac{\Sigma(\textit{recoded responses for the individual business attribute})}{\Sigma(\textit{number of recoded responses for the individual business attribute} \neq 0)}$$

$$\textit{value of business attribute} = \frac{\Sigma(\textit{value of individual business attributes})}{\Sigma(\textit{number of recoded responses for the individual business attributes} \neq 0)}$$

Release Your Refactoring Superpower

- **Case Study**

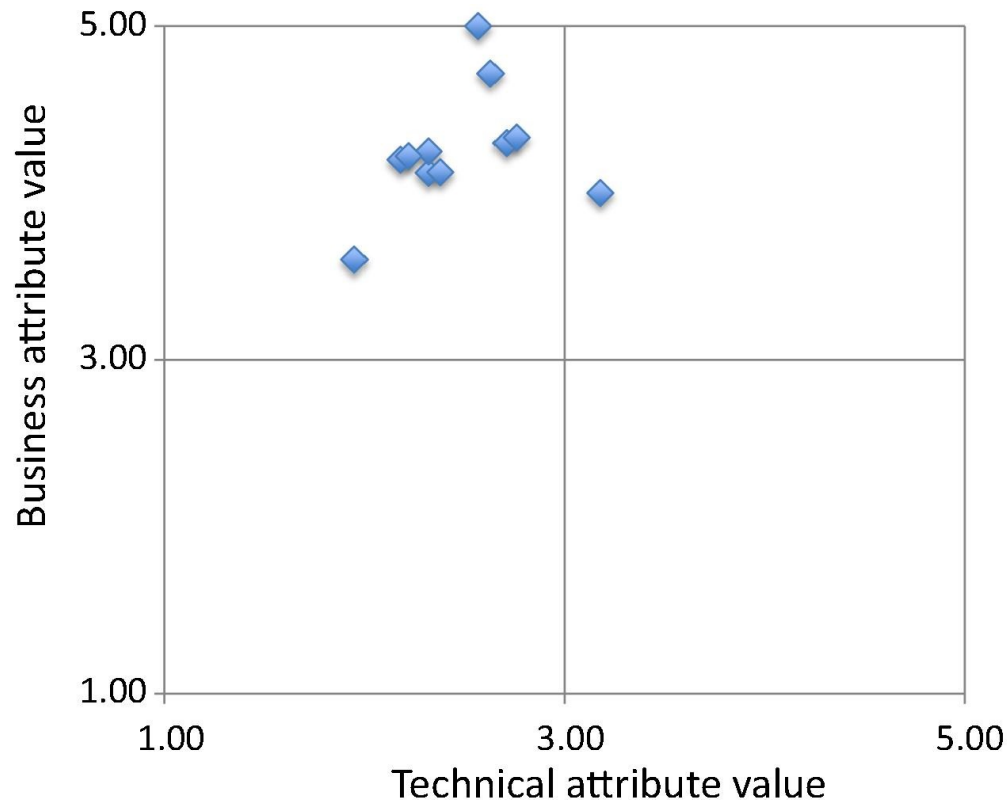
- **Example Assessment**

- **Step #6 - Plotting**

- **Display points on the decisional matrix**

Reengineer to improve quality.
Replace with COTS if available.

Expensive to maintain and low rate of return.
Decommission.



Continue normal system maintenance.

Retain while not expensive to maintain.
Decommission if expensive changes become necessary.

Release Your Refactoring Superpower

- **Superhero**
 - Status Granted
 - It's YOU!

Release Your Refactoring Superpower

- **Supervillain**

- Professor LOC

- Fighting tools

- PHPLoc

- PHPmd (codesize)

```
$ php phploc.phar -v --names "*.php" --exclude 'vendor'  
/path/to/project/my-project/module/ > /path/to/project/phpqatool-  
results/phploc.txt
```

```
$ php phpmd.phar /path/to/project/my-project/module/ xml codesize --exclude  
'vendor' --reportfile ' /path/to/project/phpqatool-  
results/phpmd_codesize_output.xml'
```

Release Your Refactoring Superpower

- The Data
 - PHPLoc

Directories	77
Files	408
Size	
Lines of Code (LOC)	139013
Comment Lines of Code (CLOC)	39849 (28.67%)
Non-Comment Lines of Code (NCLOC)	99164 (71.33%)
Logical Lines of Code (LLOC)	33765 (24.29%)
Classes	31432 (93.09%)
Average Class Length	82
Average Method Length	5
Functions	0 (0.00%)
Average Function Length	0
Not in classes or functions	2333 (6.91%)
Complexity	
Cyclomatic Complexity / LLOC	0.17
Cyclomatic Complexity / Number of Methods	2.10
Dependencies	
Global Accesses	140
Global Constants	0 (0.00%)
Global Variables	0 (0.00%)
Super-Global Variables	140 (100.00%)
Attribute Accesses	7972
Non-Static	7972 (100.00%)
Static	0 (0.00%)
Method Calls	23650
Non-Static	23299 (98.52%)
Static	351 (1.48%)
Structure	
Namespaces	60
Interfaces	0
Traits	0
Classes	379
Abstract Classes	0 (0.00%)
Concrete Classes	379 (100.00%)
Methods	5307

Release Your Refactoring Superpower

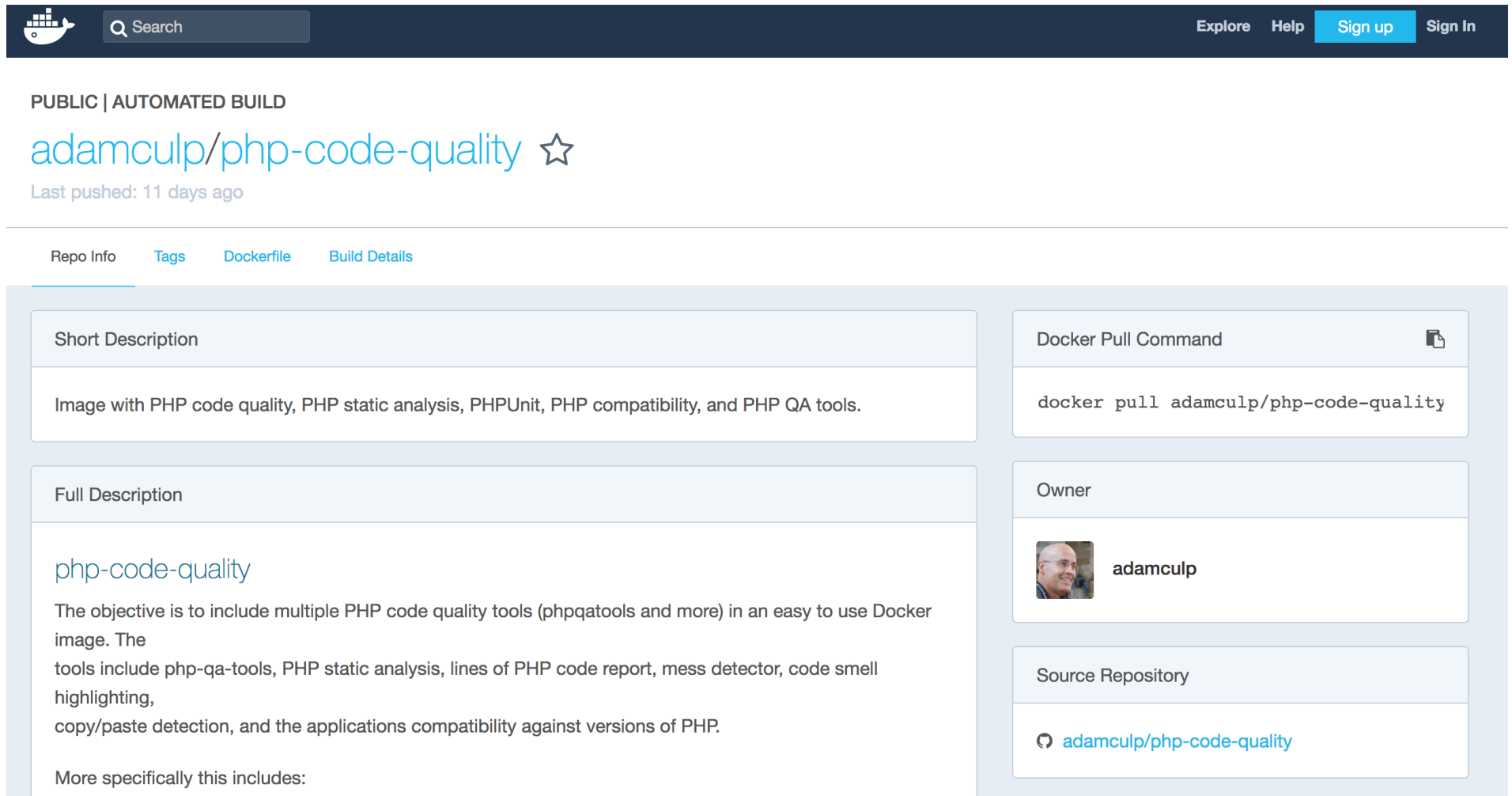
- The Data

- PHPmd

```
-<violation beginline="28" endline="1057" rule="ExcessiveClassLength" ruleset="Code Size Rules" package="Other\Controller"
externalInfoUrl="http://phpmd.org/rules/codesize.html#excessiveclasslength" class="DashboardController" priority="3">
  The class DashboardController has 1030 lines of code. Current threshold is 1000. Avoid really long classes.
</violation>
-<violation beginline="28" endline="1057" rule="TooManyMethods" ruleset="Code Size Rules" package="Other\Controller"
externalInfoUrl="http://phpmd.org/rules/codesize.html#toomanymethods" class="DashboardController" priority="3">
  The class DashboardController has 13 methods. Consider refactoring DashboardController to keep number of methods under 10.
</violation>
-<violation beginline="28" endline="1057" rule="ExcessiveClassComplexity" ruleset="Code Size Rules" package="Other\Controller"
externalInfoUrl="http://phpmd.org/rules/codesize.html#excessiveclasscomplexity" class="DashboardController" priority="3">
  The class DashboardController has an overall complexity of 143 which is very high. The configured complexity threshold is 50.
</violation>
-<violation beginline="78" endline="287" rule="CyclomaticComplexity" ruleset="Code Size Rules" package="Other\Controller"
externalInfoUrl="http://phpmd.org/rules/codesize.html#cyclomaticcomplexity" class="DashboardController" method="dashboardAction"
priority="3">
  The method dashboardAction() has a Cyclomatic Complexity of 24. The configured cyclomatic complexity threshold is 10.
</violation>
```

Release Your Refactoring Superpower

- **Ray Gun**
 - **Docker Images**



The screenshot shows the Docker Hub interface for the repository `adamculp/php-code-quality`. The page includes a search bar, navigation links (Explore, Help, Sign up, Sign In), and repository details. The repository is public and has an automated build. The short description is "Image with PHP code quality, PHP static analysis, PHPUnit, PHP compatibility, and PHP QA tools." The full description states the objective is to include multiple PHP code quality tools in an easy-to-use Docker image, listing tools like `php-qa-tools`, static analysis, code report, mess detector, code smell highlighting, copy/paste detection, and compatibility against PHP versions. The owner is identified as `adamculp`. The Docker pull command is `docker pull adamculp/php-code-quality`. The source repository is `adamculp/php-code-quality`.

Explore Help Sign up Sign In

PUBLIC | AUTOMATED BUILD

[adamculp/php-code-quality](#) ☆

Last pushed: 11 days ago

Repo Info Tags Dockerfile Build Details

Short Description

Image with PHP code quality, PHP static analysis, PHPUnit, PHP compatibility, and PHP QA tools.

Full Description

[php-code-quality](#)


The objective is to include multiple PHP code quality tools (`phpqatools` and more) in an easy to use Docker image. The tools include `php-qa-tools`, PHP static analysis, lines of PHP code report, mess detector, code smell highlighting, copy/paste detection, and the applications compatibility against versions of PHP.

More specifically this includes:


Docker Pull Command

```
docker pull adamculp/php-code-quality
```

Owner

 **adamculp**

Source Repository

 [adamculp/php-code-quality](#)

Release Your Refactoring Superpower

- **Arsenal**
 - 1 million commits
 - Rename Variable/Method/Class 77%
 - Extract Constant
 - Make Type Global
 - Rename Refactoring Command
 - Move/Extract Class 1%
 - Move/Extract Method 13%
 - Modify Method Parameters



- **Thank you!**

- Code: <https://github.com/adamculp/>

Adam Culp

<http://www.geekyboy.com>

<http://RunGeekRadio.com>

Twitter @adamculp

Questions?

