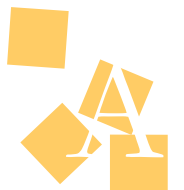


# Business Rules Implementation - Pitfalls and Lessons Learned

Michael Krouze, Alyce Neperud, and  
Krzysztof Karski

Artemis Alliance, Inc.



Artemis  
Alliance,  
Inc.

# Management Point of View

Michael Krouze, CTO  
Artemis Alliance, Inc.



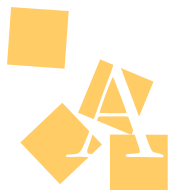
**Artemis  
Alliance,  
Inc.**

# Understanding the Costs Involved

Not all costs of Business Rules understood

Seeing it only as a technology investment or enabler

Cost overruns quickly happen due to effort not planned for



# Understanding the Costs Involved

Clearly analyze all rule costs and communicate to management

Make sure to include the following types of costs:

- Purchases
- Culture/process change
- Education
- Implementation
- Testing
- Maintenance



# Seeing the Enterprise Investment

Viewing Business Rules initiatives as a tactical, single application investment.

Ability to leverage new approach and technology investment across multiple applications and business lines is not pursued.

Business Rules not part of enterprise architecture strategy



# Seeing the Enterprise Investment

Examine all the benefits of Business Rules in the organization.

Think of Business Rules as a strategic tool for the organization

- IT agility
- Management of knowledge as an asset

Leverage the investment in tools and skills

# Providing Consistent Support

Management has not really bought in on the value of Business Rules

Support provided to the team(s) is inconsistent or weak.

Team(s) not energized to do well and typically will fail

- Self-fulfilling prophesy



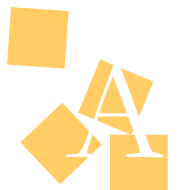
# Providing Consistent Support

Educate, review competition, review success statistics to build confidence.

Communicate strong support.

Don't vacillate – make a decision and stand behind it.

- If you decide you are wrong you can change it – just don't be weak



# One Time Only Initiative

Viewing the Business Rules project as a once-and-done project.

Not seeing the long-term benefits or costs.

Not planning for change or increased agility

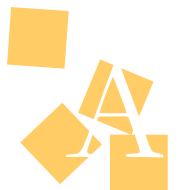


# One Time Only Initiative

Understand the real benefits of using Business Rules and how they change over time.

Recognize that this is a different way of building and managing application functionality over time.

Learn to leverage the investment in other areas of the organization.



# Short-changing the Investment

Not investing enough to get the proper return.

May cause poorer than expected results or possible project failure



# Short-changing the Investment

Understand the costs

Recognize needs for education and/or consulting support

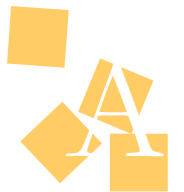
Allow for proof-of-concept work and learning curve.

Contract for support services from vendor



# Technical Point of View

Krzysztof Karski, Senior Consultant  
Artemis Alliance, Inc.



**Artemis  
Alliance,  
Inc.**

# Concept Model Changes

Rapid evolution of the concept model at beginning stages of development

External entity model changes

Refactoring of concept model as new findings are made



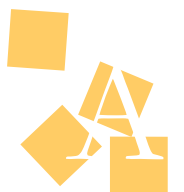
# Concept Model Changes

## Architect concept model upfront

- Reduce change by having a well thought out and architected concept model
- Have enough of a concept model architected so that future changes are additive, not structural

## Coordinate Change

- Establish a communication process between teams that might affect the concept model
- Ensure the external entity model and the concept model evolve in the same direction
- Agree early on data types, data constraints etc



# Concept Model Changes

## Evaluate impact of change

- Let them make mistakes quickly and see the value of the approach

## Delay model integration

- Isolate concept model from external entity model churn
- Figure out your rules and concept model first, then figure out how to integrate with your external entity model

# Reorganizing Rules and Rulesets

Different naming of the same concept in different rulesets or branches of the repository

Interdependency between ruleflow design and rule organization

Other “vendor specific” features attached to particular rules that cannot move

Shared resources invisible within repository scopes



# Reorganizing Rules and Rulesets

## Know your rule engine

- Understand your limitations given your vendor's strengths and weaknesses

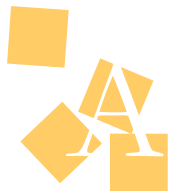
## Design upfront

- Chose an organizational paradigm and follow through
- Establish rule organization, naming and categorization standards
- Establish a vision for evolving the repository
- Architect for rule portability and compatibility across your repository

# Skipping Initial Testing

“We have an application level testing approach, we’ll test the rules there”.

“Testing rules in the authoring tool is too bothersome and different from how we test elsewhere”



# Skipping Initial Testing

## Follow the rule authoring steps

- Do not skip rule authoring steps including testing inside the rule authoring tool

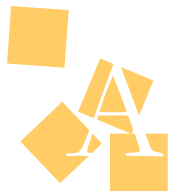
## Perform quick & targeted testing

- At least set up quick and targeted smoke tests for your rules
- Establish a reusable base of testing plumbing and base test data
- Catch 80% of mistakes in the tool
- Let the application testing strategy catch the remaining 20%

# Know Your Rule Engine

Not understanding the “correct” techniques and workflow for using the tool

Loosing out on efficiency, helpful mechanisms and streamlined workflow by not knowing the vendor’s rule authoring approach



# Know Your Rule Engine

## Know the tool

- Understand configuration options and their differences
- Use wizards and auto generation techniques where possible
- Use the appropriate data types, built in functions etc

## Understand your vendor's workflow

- Take full advantage of productivity aids by following the vendor's rule authoring workflow
- Understand the order of events when writing rules to avoid rework or unnecessary manual work

# Use the Rule Engine for Rules

Rule engines are not integration, data transformation or translation platforms

You should not have to write a ton of rules just to access, understand and return data



# Use the Rule Engine for Rules

## Integration belongs elsewhere

- Your rule application can, but should not have to, transform inputs and format outputs to make itself understood
- There are better transformation and integration tools available
- Prepare inputs outside the engine and serve them up as needed up the engine as much as you can

# See Rules as Just Code

Rules becoming as complex as code and  
incomprehensible

See rules as just a new form of writing code



# See Rules as Just Code

## Use Intermediary Concepts

- Define intermediary concepts globally and reuse them
- Define simplification concepts we make in everyday conversation and use them in your rules
- Revisit OO analysis fundamentals and apply them

## Follow rule writing best practices

- Have each rule deal with one issue
- Break up large decision logic blocks into many rules
- Don't write techie rules

# Analysis Point of View

Alyce Neperud, Principal  
Artemis Alliance, Inc.



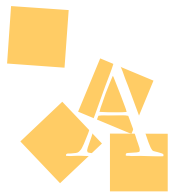
**Artemis  
Alliance,  
Inc.**

# Resistance to Change

“We know what Business Rules are so we’ll do them the way we want.”

Not wanting to adopt formal methods.

No belief in the value of a Business Rules approach.



# Resistance to Change

## Educate more

- Give better understanding of why we are doing it this way

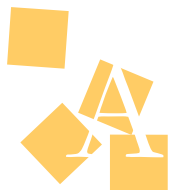
## Build support within all areas

- Development, Analysis, Testing, Project Management

## Enable them to “skin their knees”

- Let them make mistakes quickly and see the value of the approach

Bring understanding of the “big picture” and the value of the approach down-stream in the development process.



# Separation of Rules & Requirements

Tricky to understand the separation

Use Case Writers and Rules Writers not communicating

- Results in rules captured in requirements documents and functional requirements being captured as rules



# Separation of Rules & Requirements

Teach them to write good requirements and good rules

Use concrete examples to demonstrate

- Show the new version of artifacts after introducing the concrete examples

Use short, fast iterations with experienced review to provide experience and feedback

# Poor Repository Planning

Lack of proper forethought into what would be required to effectively manage changes.

Assuming that not much would be required in a tool.

Document-centric approach

- Too much time spend worrying about where to put the rule in the document as opposed to making sure it's a good rule.

# Poor Repository Planning

Use knowledgeable people in designing the rules repository.

Work through the versioning and change management in detail.

Execute full rule lifecycle testing of the repository.

# Not Everyone Makes a Good Rules Analyst

Assuming that everyone would be a good analyst.

People may just write rules without any proper business analysis

- They look like good rules but don't really capture the business need.

# Not Everyone Makes a Good Rules Analyst

Don't assign critical tasks to people who are unproven.

Have internal or external mentors who have time to help.

Pair experienced people with inexperienced.

Explicitly determine skill levels of team members



# Solving Same Problem in Different Ways

“Too many cooks in the kitchen”

Lack of communication

Poor task division



# Solving Same Problem in Different Ways

Establish model for sharing information and lessons learned.

Don't overwork your experts.

Be careful in how you divide task efforts so that you don't assign people with overlapping work.



# No Champion

The project does not have a rules champion with authority.



# No Champion

Get one!



**Artemis  
Alliance,  
Inc.**

Thank You!

Artemis Alliance, Inc.



**Artemis  
Alliance,  
Inc.**