

# Leveraging CA Gen's platform strengths

Tim Dargavel, Facet Consulting

CA Gen user group, Hamburg  
Wednesday 26<sup>th</sup> March, 2014.

delivering **strategic** enterprise **solutions**



# Who are we?



Global CA Gen services specialists

Upgrades and Performance tuning

Application Development and Maintenance

Application Transformation and Modernization

Application Architecture and Strategic reviews

CA Gen training and Architecture Consulting

Offices in Australia and the United Kingdom

A global CA Technologies services partner.

## This afternoon



### CA Gen's **strategic strength is platform independence**

- ✓ Isolates Developers from the underlying technology
- ✓ Enables applications to be deployed to new platforms

Customers **can** leverage this strength to move platforms

- ❑ A set of steps for approaching this activity
- ❑ Areas to focus on when scoping the activity
- ❑ A recommended approach with actions to take

Customers are successfully using this approach right now.

## 3 Stepping stones

1. Understand why you need to move platform

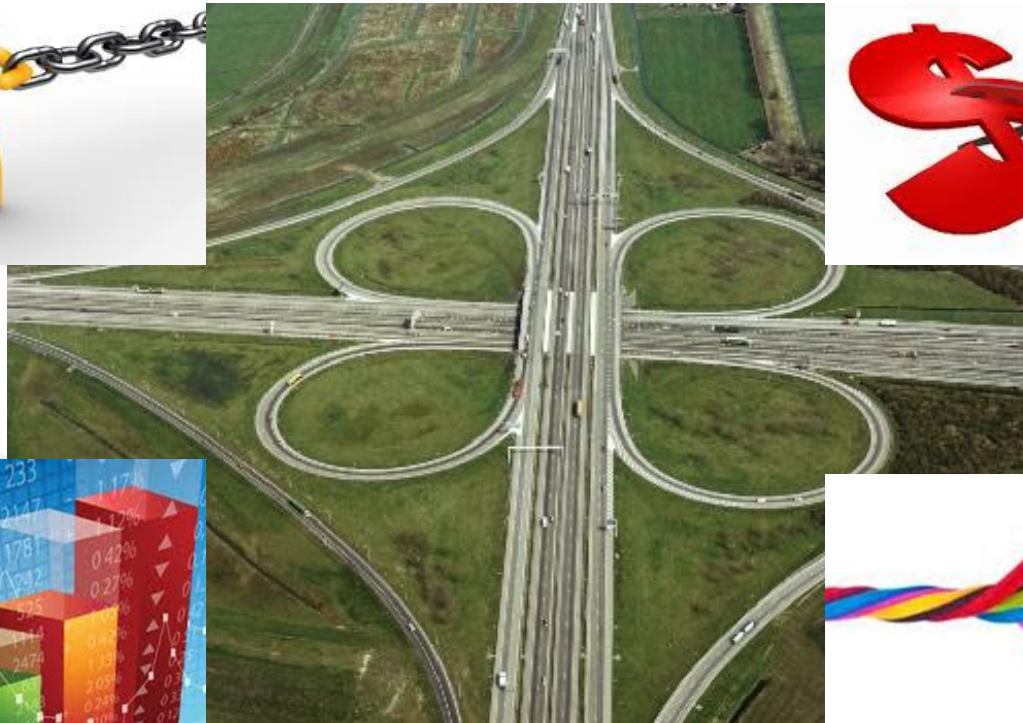


# So why change platform?

## Constraints



## Costs



## Capacity



## Convergence

# Drivers define the business case



## Business related

## IT focused

Constraints



Costs

Capacity



Convergence

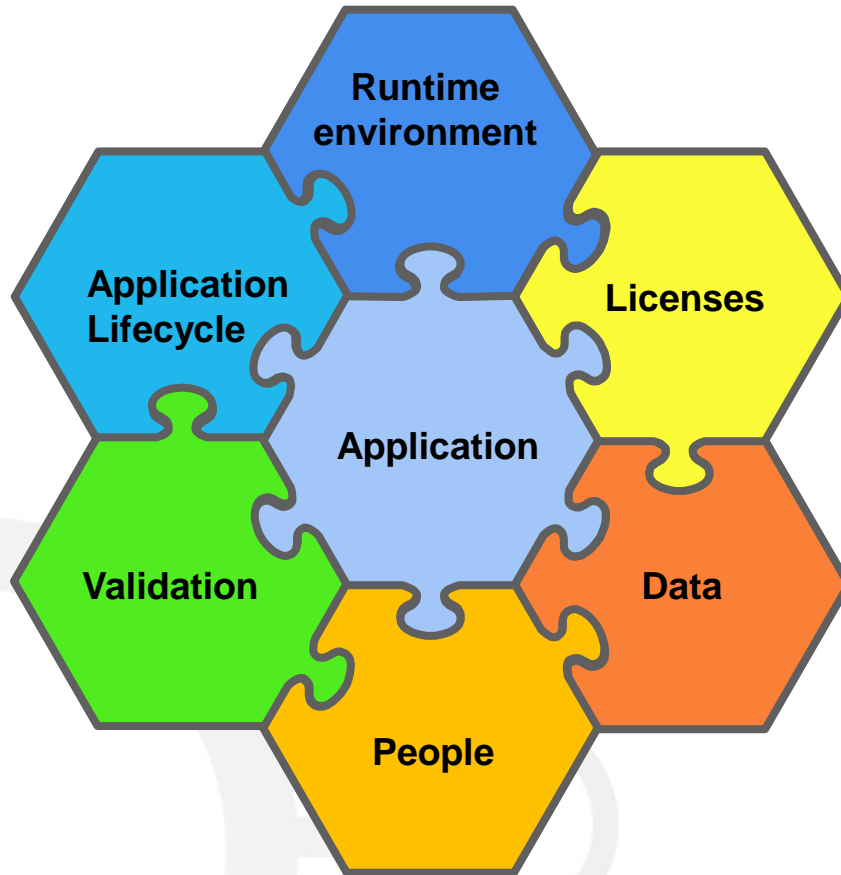
# Sponsors will validate your project drivers.

## 3 Stepping stones

1. Understand why you need to move platform  
***This defines your business case and sponsors***
2. Define the scope of what would be affected



# Qualify and quantify scope



Application(s) & Portfolio

Runtime environment

Software Licenses

Data – content and location

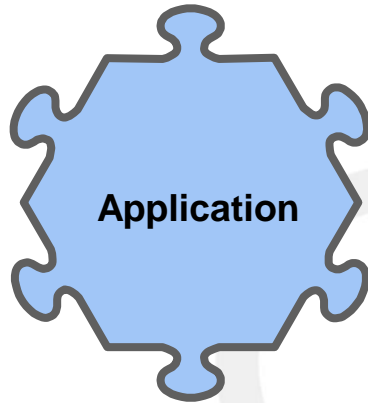
People – skills & experience

Validation – artifacts & project

Lifecycle – Processes & tools.



# Qualify and quantify scope



## Application(s)

- For each application, what is the
  - Size – Source & Transaction volume
  - Criticality – Business & Technical SLAs
  - Volatility – Quantity of ongoing change
  - Diversity – CA Gen and non-CA Gen
- What else is in the application portfolio?
- What external integration points exist?

# Qualify and quantify scope



## Runtime environment

- What is the environmental software stack?
  - Operating system
  - Database & TP Monitor platforms
  - Messaging, Security & Batch
- What application monitoring is in place?
- What are the application “hooks”?
- What customizations have been made?

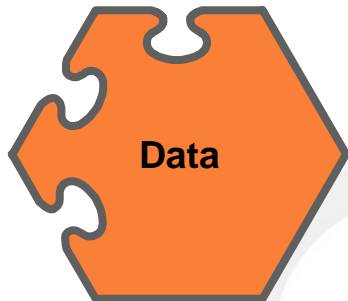
# Qualify and quantify scope



## Software licenses

- Which ones do you need to keep?
  - Vendor, product and cost?
  - Current contract durations?
- What new licenses will you need?
  - What licenses can you leverage?
  - What vendors do you already have?
  - What are purchasing processes?

# Qualify and quantify scope



## Data – content, location and access

- What data sources are used?
- Do other applications use those sources?
- Does the data need to move platform?
- How else is the data accessed?
- What are the backup & recovery impacts?

# Qualify and quantify scope



## People – skills and experience

- Do you have skills on the new platform?
- Can you leverage existing skill sets?
- What happens to existing people?
- What training will your developers need?
- A critical success factor – address the FUD.

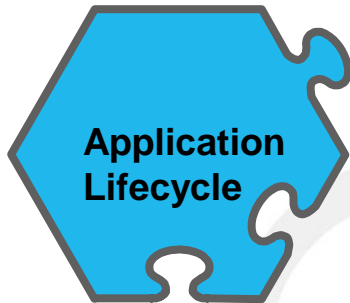
# Qualify and quantify scope



## Validation – does it still work?

- How is application change validated now?
- Will that approach still work going forward?
- Can that approach be used for the project?
- Do tools in use work on the new platform?
  - Formal testing tools
  - Development group scripts & utilities
- How will you prove you're successful?

# Qualify and quantify scope



## Application lifecycle – tools and processes

- Will the current Encyclopedia be retained?
- What source code controls will be used?
- Will configuration management change?
- How will applications be deployed?
- Which processes will need to change?
- Who retains which authorities in releases?

## 3 Stepping stones

1. Understand why you need to move platform  
*As this defines your business case and sponsors*
2. Define the scope of what would be affected  
***So you can identify stakeholders and define effort***
3. Start with a rapid proof-of-concept





# Proving the concept



## Objective

A working “sliver” of your application on the new platform

## Requires

Standing up a small scale target platform

Identifying a discrete application sliver to regenerate

Creation of the database and migration of the required data

Deployment of the new application runtimes

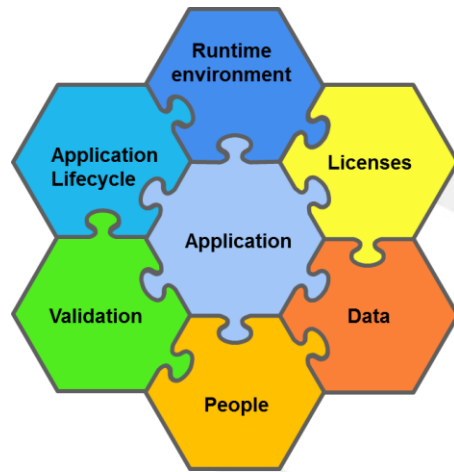
## Outcomes

Definition of where the “pain points” are

Exploration of what else is required to scale

Objective data that quantifies the business case and risks.

# Proving the concept



Scope the Application

Build the Runtime environment

Obtain the Licenses

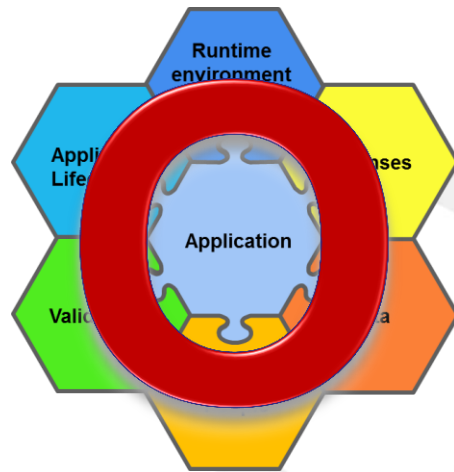
Migrate the Data

Identify the People

Determine how to Validate

Cover your Lifecycle processes.

# Proving the concept - Scope



## Scope the Application

Needs to be representative of complexity

Needs to be reasonably self-contained

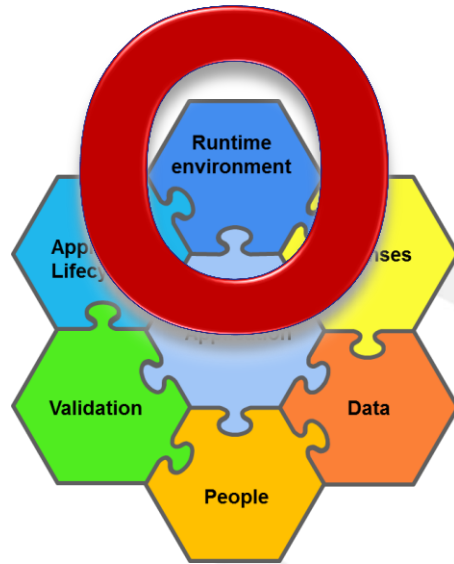
Needs to utilize self-contained data

Needs to deliver the principle UI

Include a small number of EABs

Caution on external integration.

# Proving the concept - Runtime



## Build the Runtime environment

Has to be operational - but not production

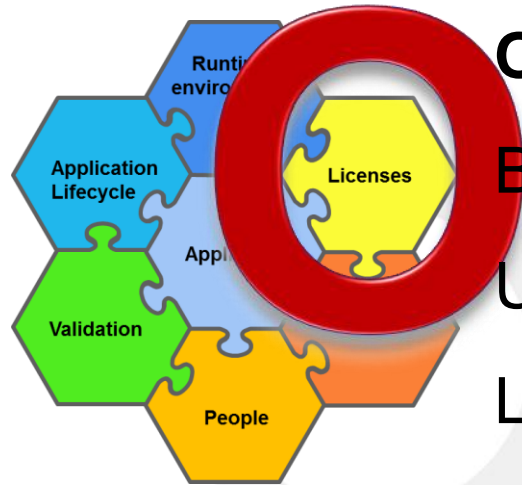
Utilize, or build as, an isolated “sandbox”

Emphasize operational over scalable

Ideally, use supported software versions

- Compilers
- Databases
- Web and Application servers, etc.

# Proving the concept - Licenses



## Obtain the software licenses

Beg and borrow what's necessary

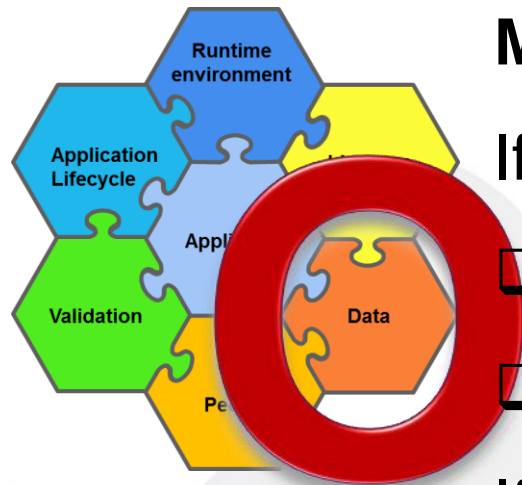
Use trial licenses from vendors

Leverage existing Enterprise licenses

Explore supported Open Source options

Use the “consumer versions” for POCs.

# Proving the concept - Data



## Migrate the data

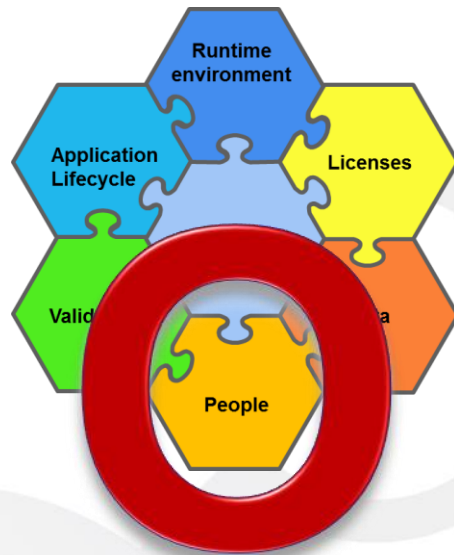
If the database server is separate, then

- Understand concurrent access
- Identify the appropriate database

If the database will be ported also, then

- Identify skills to stand up the software
- Generate the DDL for the database
- Identify your data transfer mechanism.

# Proving the concept - People



## Identify the People

You'll need different infrastructure skills

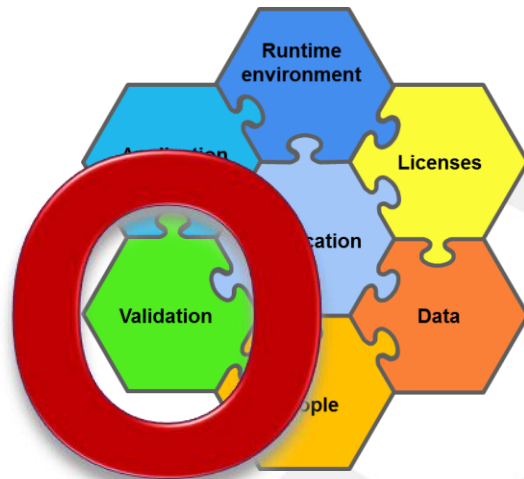
- Operating system
- Software installation and configuration
- Databases and Application servers
- EABs and source code control

Borrow the skills you need – short term

Find those skills within your own teams

**The right people make ALL the difference!**

# Proving the concept - Validation



## Work out how to Validate

For the POC application sliver

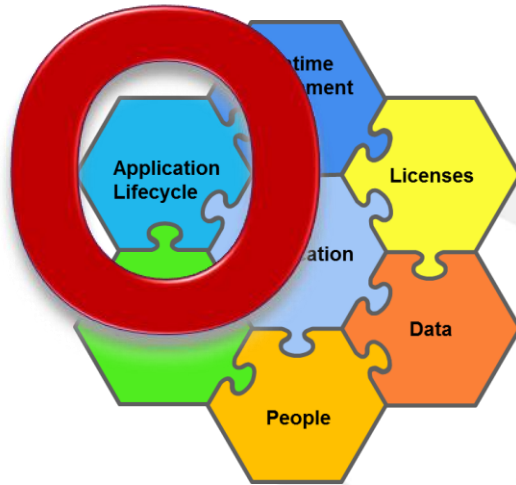
- What test scripts can be re-used
- What database matching can be done

What constitutes success for the POC

- Functional application results?
- Underlying data integrity?
- Performance, integration, load levels?
- User interface equivalence/standards?



# Proving the concept - Lifecycle



## Cover your Lifecycle processes

Cut a separate set of models

Potentially, build a new Encyclopedia

Install supporting tools you currently use

Identify and trial the new tools you need

Define the new source code management

Execute an application change cycle.

## Proving the concept

- ✓ The “working” application exists only to prove the concept
  - ❑ Do **NOT** expect to utilize the sandbox as Production
  - ❑ Emphasize workable over ideal
  - ❑ *Quick and Dirty* is ok... but keep track of the shortcuts you take
- ✓ Deliver the data to quantify your business case
  - ❑ A pound of walk is worth a ton of talk
  - ❑ An outcome you can point to answers questions and doubters
- ✓ A successful POC is only the start of the journey.

## 3 Stepping stones

1. Understand why you need to move platform  
***As this defines your business case and sponsors***
2. Define the scope of what would be affected  
***So you can identify stakeholders and define effort***
3. Start with a rapid proof-of-concept  
***Provide the data to make informed decisions.***



## Recommendations

1. No “sacred cows” – ask the obvious/difficult questions
  - ❑ “The mainframe is more expensive” - Compared to what?
  - ❑ “It’ll be quicker to rewrite” – How long, how much, compared to?
  - ❑ “What really happens if we do nothing?” – The default state
2. Identify all the drivers to identify your sponsors
3. Use a short and sharp POC – 1-3 folks for 1-3 months
4. Use POC iterations to expand scope and learn lessons
5. Each iteration justifies proceeding further, or stopping.

**Thanks!**



For **copies** of this presentation or to **discuss further**

tim.dargavel@facet.com.au

+44 7798 523-857 (UK)

+61 417 009-311 (AU)

+61 7 3010-6001 (direct)

Find out **more about us**

www.facet.com.au

<http://www.ca.com/us/collateral/service-partners/na/Facet-Consulting-Partner-Spotlight.aspx>