Measure Your Agile Architecture Maturity

The RCDA Maturity Model

Eltjo Poort
O’Reilly Software Architecture Conference, Berlin
November 2019
Eltjo Poort
CGI

RCDA
Founder / owner
CGI’s agile architecture approach

PhD
Improving Solution Architecting Practices

Architecture
Practice Lead
Reviewer
Coach
Agile transformer

14 Peer-reviewed pubs
2 Best paper awards
3 Best presn awards

2016
Linda Northrop Architecture Award

http://eltjopoort.nl
Not another maturity model…

What is a maturity model?
Capability maturity models focus on improving processes in an organization. They contain the essential elements of effective processes for one or more disciplines.¹

• The benefits of defined processes in software engineering have been disputed, but establishing good practices is useful.²

Why measure agile architecture maturity?
• To identify areas of improvement
• To track improvement progress

1 SEI (2003), CMMI: Guidelines for Process Integration and Product Improvement
Foundation

Risk and Cost Driven Architecture

- Agile solution shaping, quick feedback loop
- Focus on economic reality
- Scalable architecting
- Better risk and cost control in delivery

RCDA publications:

- Journal of Systems and Software
- Leading architecture conferences: SATURN, LAC, WICSA (best paper award)

Inspiration

What do architects actually do?

- Architecting: design, validation, prototyping, documenting, etc.
- Getting input: user, requirements, other architecture, technology
- Providing information: communicating architecture, assisting other stakeholders

Eoin Woods
CTO at Endava

- Consider external dependencies
- Look for novel aspects of the domain, problem or solution
- Identify the high impact decisions
- Analyze your local situation for risks

- Understand stakeholder needs and priorities
- Prioritize time according to risks
- Delegate as much as possible

- Consider the whole stakeholder community
- Ensure the needs of the delivery team are understood and met
- Understand the perspective of the acquirers ("The Business")
- Empower development teams (architecture as a role & activity not a person)
- Create groups to take architectural decisions
History of digital (software) architecture

Fred Brooks
Conceptual integrity
1968

Mary Shaw
David Garlan
Architecture as high-level abstractions
1995

Jansen & Bosch
Tyree & Akerman
Architecture as design decisions
2000

Poort & van Vliet
Risk and cost driven architecture
2005
2010
2011

Edsger Dijkstra
Software structure matters
1975

Philippe Kruchten
4+1 Views for architecture documentation
1996

IEEE 1471
Standard for architecture documentation
2005

George Fairbanks
Risk-driven architecture

What
How
Why
The Five Responsibilities of the Architecture Function

Understanding Context

Making Decisions

Modeling

Validating

Fulfillment

Architecture as a set of design decisions
Tyree, Bosch, Kruchten, Woods

Architecture is context

Architecture as an abstraction
Shaw, Garlan

Architecture as a set of structures
SEI, Kruchten, Rozanski & Woods

Architecture as a risk and cost management discipline
Fairbanks, Poort
The Waterfall Wasteland

“Making Decisions" - "We don’t take decisions, we only advise management”

“Validating” - “Our design was perfect, but the builders were incompetent”

“Modeling”
The Agile Outback

“The best architectures emerge”

“The best architectures emerge”

“Fail early and fail often”

“Fail early and fail often”

Understanding Context

Making Decisions

Fulfillment

Validating

Modeling
An organization’s architecture function is mature if:

- It pays balanced attention to all five responsibilities
- Activities in the five responsibility areas are coherent and related to each other

Organizations can be mature irrespective of how the architecture function is organized, or what it is called (roles, teams, boards,…).
Mapping responsibilities to behavior

Understanding Context

Effective stakeholder communication
Context knowledge managed

Visual model of context
Visual models of solution

Modeling

Making Decisions

Decisions as primary deliverable
Prioritized by business impact
Justified and documented
Well-timed decisions
Decentral unless…

Validating

Fulfills stakeholder needs?

Fulfillment

Architectural runway recognized
Architecture debt control
Just enough anticipation
Measuring behavior

- Interviews with 5 architects and stakeholders per team
- Asking open questions
- Referencing examples of evidence and counter-evidence

Behaviors scored 1-5:
1 Never
2 Seldom
3 Ad hoc/Individuals
4 Mostly
5 Habit

<table>
<thead>
<tr>
<th>Evidence</th>
<th>Counterevidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business stakeholders with sufficient mandate are actively involved in architecture activities</td>
<td>Business stakeholders are only involved at pre-defined approval moments. Business stakeholders are only involved in requirements gathering.</td>
</tr>
<tr>
<td>Delivery and operational stakeholders are actively involved in architecture activities</td>
<td>The architecture is only based on business concerns, delivery and operational concerns are left to others or too late.</td>
</tr>
<tr>
<td>Architects communicate with business stakeholders in business terms (business cases, risk analyses)</td>
<td>Architects use modeling language (UML, Archimate) or design pattern language (layers, APIs) to talk to business stakeholders</td>
</tr>
<tr>
<td>Architecture documentation only describes how the architecture addresses the relevant stakeholder concerns at each point in time.</td>
<td>Architecture documentation contains all knowledge gathered up till now. Architecture documentation includes viewpoints addressing concerns that may become relevant later in time.</td>
</tr>
<tr>
<td>Architectural requirements and concerns are discussed with all stakeholders in terms of scenarios (epics, stories)</td>
<td>Architectural concerns are discussed in vague terms like &quot;security&quot;, &quot;performance&quot;, &quot;velocity&quot;.</td>
</tr>
<tr>
<td>Architects have access to all stakeholders when necessary</td>
<td>Architects are not allowed to talk to business stakeholders</td>
</tr>
<tr>
<td>Architects actively look for (business) goals and drivers behind the problem they have been asked to address.</td>
<td>Architecture is based on a fixed set of documents (requirements, specs) without full awareness of the underlying business drivers.</td>
</tr>
<tr>
<td>Specific architectural drivers and the (business) goals behind them are documented and validated.</td>
<td>No description of the architectural context is made, or it is made but not validated with stakeholders.</td>
</tr>
<tr>
<td>Information system context is documented and</td>
<td>No context diagrams are made.</td>
</tr>
</tbody>
</table>
**Agile Architecture Maturity Radar**

**Understanding Context**
- Effective stakeholder communication
- Context knowledge managed

**Modeling**
- Visual model of context
- Visual models of solution

**Validation**
- Fulfills stakeholder needs?

**Making Decisions**
- Decisions as primary deliverable
- Prioritized by business impact
- Justified and documented
- Well-timed decisions
- Architectural runway recognized
- Architecture debt control
- Just enough anticipation

**Delivery**
- 1-100% Quality multiplier

<table>
<thead>
<tr>
<th>Scope/team:</th>
</tr>
</thead>
</table>

- 1 Never
- 2 Seldom
- 3 Ad hoc/Individuals
- 4 Mostly
- 5 Habit

© 2019 CGI Inc.
Agile Architecture Maturity Radar

Scope/team: WOSP Product Team

Understanding Context
- Effective stakeholder communication: 4
- Context knowledge managed: 2

Modeling
- Visual model of context: 3
- Visual models of solution: 4

Validation
- Fulfills stakeholder needs?: 5

Making Decisions
- Decisions as primary deliverable: 4
- Prioritized by business impact: 3
- Justified and documented: 2
- Well-timed decisions: 2
- Architectural runway recognized: 2
- Architecture debt control: 3
- Just enough anticipation: 2

Delivery
- Decentral unless...: 4

Effective stakeholder communication: 4
Context knowledge managed: 2
Visual model of context: 3
Visual models of solution: 4
Fulfills stakeholder needs?: 5

Agile Architecture Maturity Radar

1 Never  2 Seldom  3 Ad hoc/Individuals  4 Mostly  5 Habit  1-100% Quality multiplier
Understanding Context
Some observations

• Written documentation is never enough to fully understand context.
  • → Talk to stakeholders!
  • make sure you know who they are (all of them)

• Architectural trade-offs can only be made if the goals behind stakeholder needs are understood.
  • → Ask why! (7 times?)

• Vocabulary mismatch is a common source of misunderstanding
  • → Ask for narratives (stories, epics,…) (by asking why)
  • → Speak your stakeholders’ language
## Understanding Context

### Effective stakeholder communication

<table>
<thead>
<tr>
<th>Description</th>
<th>Evidence</th>
<th>Counterevidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architects communicate effectively with stakeholders</td>
<td>Business stakeholders with sufficient mandate are actively involved in architecture activities</td>
<td>Business stakeholders are only involved at pre-defined approval moments. Business stakeholders are only involved in requirements gathering.</td>
</tr>
<tr>
<td><strong>Delivery and operational stakeholders are actively involved in architecture activities</strong></td>
<td><strong>The architecture is only based on business concerns, delivery and operational concerns are left to others or too late.</strong></td>
<td></td>
</tr>
<tr>
<td>Architects communicate with business stakeholders in business terms (business cases, risk analyses)</td>
<td>Architects use modeling language (UML, Archimate) or design pattern language (layers, APIs) to talk to business stakeholders</td>
<td></td>
</tr>
<tr>
<td>Architectural requirements and concerns are discussed with all stakeholders in terms of scenarios (epics, stories)</td>
<td>Architectural concerns are discussed in vague terms like &quot;security&quot;, &quot;performance&quot;, &quot;velocity&quot;.</td>
<td></td>
</tr>
<tr>
<td>Architects have access to all stakeholders when necessary</td>
<td>Architects are not allowed to talk to business stakeholders</td>
<td></td>
</tr>
</tbody>
</table>
Understanding Context

Context knowledge managed

<table>
<thead>
<tr>
<th>CX_DOC</th>
<th>Knowledge about architectural context is gathered, validated and preserved</th>
<th>Architects actively look for (business) goals and drivers behind the problem they have been asked to address.</th>
<th>Architecture is based on a fixed set of documents (requirements, specs) without full awareness of the underlying business drivers.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specific architectural drivers and the (business) goals behind them are documented and validated.</td>
<td>No description of the architectural context is made, or it is made but not validated with stakeholders.</td>
<td></td>
</tr>
</tbody>
</table>
Making Architectural Decisions
Your primary deliverable

Say goodbye to The Architecture Document (as your primary deliverable)
- Takes weeks or months to produce
- Forces approval of all decisions in one go

Say hello to The Architectural Decision (as your primary deliverable)
- Finer granularity of artifact
- Easier to speed up feedback cycle
- Allows individual decision timing
Making Architectural Decisions
Decisions as primary deliverable

<table>
<thead>
<tr>
<th>AD_PRIM</th>
<th>Architectural decisions are a primary architectural deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Architectural decisions are communicated individually, whenever they are taken (e.g. collectively (implicitly or explicitly) as part of in an architectural decision register)</td>
</tr>
<tr>
<td></td>
<td>as part of a large architecture document.</td>
</tr>
<tr>
<td></td>
<td>Stakeholder feedback is gathered on individual architectural decisions</td>
</tr>
<tr>
<td></td>
<td>Stakeholders only review the collective architecture document.</td>
</tr>
</tbody>
</table>
Making Architectural Decisions
The architecture microcycle

Architectural decisions

Identify & prioritize architectural concerns

Architectural concerns (backlog)

This backlog should be prioritized by business impact (mostly risk and cost)

Decide best fitting strategy

Research possible strategies

© 2019 CGI Inc.
## Making Architectural Decisions
### Prioritized by economic impact

<table>
<thead>
<tr>
<th>AD_PRI0</th>
<th>Architectural work is prioritized by economic impact.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In the architectural workload, those concerns and decisions that have the highest risk and cost impact on their collective stakeholders are considered first, and receive the most attention.</td>
</tr>
<tr>
<td></td>
<td>Architecture work is based on / prioritized by what is mentioned in a documentation template, project plan or job description. Architectural decisions are only made when stakeholders ask for them.</td>
</tr>
<tr>
<td></td>
<td>Stakeholders have no clue about the relevance of architectural concerns.</td>
</tr>
</tbody>
</table>

Architectural concerns and their priorities are clear to stakeholders.
Making Architectural Decisions Justified and documented

- Include rationale, decision criteria, consequences, drawbacks and alternatives not chosen
- Easily accessible to stakeholders
- Let everyone know you welcome feedback
# Making Architectural Decisions

## Justified and documented

<table>
<thead>
<tr>
<th>AD_DOC</th>
<th>Architectural decisions are justified and documented</th>
<th>Architectural decisions, including their status and justification, are visible to stakeholders.</th>
<th>Stakeholders only see the effect of decisions after they have been taken in a set of views or models.</th>
</tr>
</thead>
</table>

The criteria on which each decision is based, and their relation to specific architectural drivers and business goals, are visible to stakeholders.

Justification for decisions is not documented, or is mainly based on generic principles and not related to the specific context.

Alternatives for a decision and the reason for their rejection are visible to stakeholders.

Only the chosen alternative is documented.

Architects take delivery and operational consequences of decisions into account and document them.

The target architecture is chosen, how to deliver it is left to others. Only the decision is documented, not its impact.
Architectural Decision Timing

Timing architectural decision is balancing risk, cost and delivery time:
- too little information → risk of not meeting key requirements
- waiting too long → project delays, wasted resources
<table>
<thead>
<tr>
<th>AD_TIME</th>
<th>Architectural decisions are taken at the optimal time</th>
<th>The timing of each architectural decision is a conscious process.</th>
<th>Timing of architectural decisions is dictated by project milestones or (agile) principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A decision is postponed until more knowledge is available if the risk of a wrong decision outweighs the cost of delay.</td>
<td>A decision is made now because the deadline for the architecture document is getting close.</td>
<td></td>
</tr>
</tbody>
</table>
Making Architectural Decisions

Decentral unless...

- "Emerge" does not imply "without thinking", "for free" or "magically".
- It does stimulate re-thinking ownership of architectural decisions.

Classic architecture: broad standardization

- Hard rules
  - Responsibility: Architecture
  - Left open

Evolutionary architecture: eventual integrity

- Hard rules
  - Suggestions
  - Left open
  - Responsibility: Developers
  - Responsibility: Architecture

Source: Stephan Toth
Making Architectural Decisions
Decentral unless…

Crowd-sourced

No ‘named’ architect
Architecture marshalls
Architecture owner
Classic architect

Centralized

- Project size
- Different locations
- Domain expertise

- Architecture base
- External dependencies
- Familiarity, experience

- Discipline
- Organizational context

Source: Stephan Toth
## Making Architectural Decisions

### Decentral unless...

<table>
<thead>
<tr>
<th>AD_DLGT</th>
<th>Architectural decisions are delegated to the optimal level of decentralization</th>
<th>An architectural decision is taken at the central level (only) if decentral interests can be conflicting or cause complexity that outweighs local benefits</th>
<th>All topics mentioned in the reference architecture template are made at the central level.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An architectural decision is taken at the decentral level (only) if the benefits of local optimization outweigh the risk and costs of diversity and non-standardization</td>
<td>All decisions are made at the decentral level because we are a self-organizing team applying agile principles.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Responsibility for architectural decisions is shared between architects owning the wider context and architects owning specific solution (elements).</td>
<td>Architects demand a clear delineation of their own responsibility.</td>
<td></td>
</tr>
</tbody>
</table>
Context Diagram: Solution in its operational environment

- “What’s in scope and what is not?” → Solution Boundary
- “What external systems/actors?” → Interface Overview

## Architecture Modeling

### Visual model of context

<table>
<thead>
<tr>
<th>MD_CX</th>
<th>There is a visual model of the solution Information system context is documented and validated (context diagram), showing solution boundary and external dependencies.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No context diagrams are made.</td>
</tr>
</tbody>
</table>
Architecture Modeling
Visual models of solution

All architecture documentation methods use **views**

- ISO 42010, TOGAF, Archimate, 4 + 1, ‘Views and Beyond’, C4

- Viewpoints address concerns per stakeholder (group)
- Concerns evolve over time, use only relevant views at each time

![Diagram](https://c4model.com)

Philippe Kruchten

![Diagram](https://en.wikipedia.org/wiki/4+1_architectural_view_model)

Simon Brown

© 2019 CGI Inc.
## Architecture Modeling

### Visual models of solution

<table>
<thead>
<tr>
<th>MD_SL</th>
<th>Appropriate visual models of the solution are created</th>
<th>Visual models are created that show how the architecture addresses the relevant stakeholder concerns at each point in time.</th>
<th>Architecture documentation contains all knowledge gathered up till now. Architecture documentation includes viewpoints addressing concerns that may become relevant later in time.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Visual models are used to validate that key stakeholder needs are fulfilled by the design.</td>
<td>Diagrams are only used to clarify written documentation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Visual models are used as the basis for creating and delivering the solution.</td>
<td>Models are used to get approval and then abandoned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Visual models are made at the appropriate level of abstraction to allow reasoning about the architecture.</td>
<td>Diagrams are created at prescribed levels of abstraction only.</td>
</tr>
</tbody>
</table>
### Architecture Validation

**Fulfills stakeholder needs?**

<table>
<thead>
<tr>
<th>VL_ANLYS</th>
<th>The architect/team checks whether the architecture (still) fulfills the stakeholder needs.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Architecture models and checklists are used to validate the architecture against stakeholder needs.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>We'll first build the architecture and then test it.</strong></td>
</tr>
<tr>
<td></td>
<td>Architectural requirements, such as non-functional requirements, are made part of the &quot;definition of done&quot; and are tested.</td>
</tr>
<tr>
<td></td>
<td>Only functionality is tested.</td>
</tr>
<tr>
<td></td>
<td>There are performance/security tests because they are mandatory.</td>
</tr>
</tbody>
</table>
Architecture Fulfillment
Architectural runway recognized

<table>
<thead>
<tr>
<th>Positive Value</th>
<th>Negative Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Visible</strong></td>
<td><strong>Invisible</strong></td>
</tr>
<tr>
<td>New features</td>
<td>Architecture Runway</td>
</tr>
<tr>
<td>Added functionality</td>
<td></td>
</tr>
<tr>
<td><strong>Defects</strong></td>
<td>Technical Debt</td>
</tr>
</tbody>
</table>

Source: Philippe Kruchten

What's in your backlog?
(or Work Breakdown Structure / Project Portfolio / Change Requests)
Architecture Fulfillment

Architectural runway recognized
**Behavior**

<table>
<thead>
<tr>
<th>FF_AR</th>
<th>Work on architectural runway is recognized and planned.</th>
</tr>
</thead>
</table>

**Description**

- The product backlog or product breakdown structure contains stories or activities to realize architectural elements and technical debt reductions ("enablers") whose business value is indirect (i.e. they derive their business value from other improvements that depend on them).
- Ownership for architectural decision making is clearly allocated (to an individual or team).
- Improvements considered for the next release/sprint are fed by a variety of sources, among which is the architect(ure team).

**Evidence**

- The product backlog only contains stories that add direct business value.
- The work breakdown structure only contains work items to realize functionality.
- Only new functionality or features are considered for the next release/sprint.
Architecture Fulfillment
Architecture Debt Control – Business Case

Make Architectural Debt visible as **business risk**
- Put on risk register
- Find business owner(s) who feel the pain of the risk (and can do something about it)
- Make the business case

<table>
<thead>
<tr>
<th>Item</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Benefits</strong></td>
<td></td>
</tr>
<tr>
<td>Reduced recurrent maintenance cost</td>
<td>M/yr</td>
</tr>
<tr>
<td>Reduced risk exposure</td>
<td>R/yr</td>
</tr>
<tr>
<td>Total benefits per year</td>
<td>M+R</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td></td>
</tr>
<tr>
<td>Principal: effort of migration/refactoring/…</td>
<td>P</td>
</tr>
<tr>
<td>Opportunity cost (delayed features)</td>
<td>F</td>
</tr>
<tr>
<td>Total cost</td>
<td>P+F</td>
</tr>
<tr>
<td><strong>TOTAL RETURN ON INVESTMENT (1 YEAR)</strong></td>
<td>(M+R) – (P+F)</td>
</tr>
</tbody>
</table>
### Architecture Fulfillment

#### Architecture Debt Control

<table>
<thead>
<tr>
<th>FF_TD</th>
<th>Architectural debt is identified and controlled.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Technology upgrades and work to fix architectural shortcuts are visible on the product backlog or project schedule.</td>
</tr>
<tr>
<td></td>
<td>The product backlog only contains stories that add direct business value. The work breakdown structure only contains work items to realize functionality.</td>
</tr>
<tr>
<td></td>
<td>Decisions to fix architectural debt are based on economic trade-offs taking into account interest and cost of delay.</td>
</tr>
<tr>
<td></td>
<td>Architectural debt is only fixed when there is time left after implementing all the necessary functionality.</td>
</tr>
</tbody>
</table>
Architecture Fulfillment
Just enough anticipation

Just Enough Anticipation achieved by:

• Dependency Analysis
• Technical Debt Control
• Economic Reasoning

Source: Nanette Brown, Rod Nord, Ipek Ozkaya

Legend
- User feature
- Architectural improvement
- Defect removal
- Technical debt reduction

© 2019 CGI Inc.
Architecture Fulfillment

Strategy 1: Value-first roadmapping

- In line with Agile philosophy
- May increase TCO (more refactoring)
- Too “greedy” algorithm may run project into wall (complete rebuild)
- Good in volatile environments
Strategy 2: Architecture-first roadmapping

- In line with plan-driven philosophy
- Late delivery of value → risk of cancellation
- Risk of building wrong architecture (if context changes)
- Good for complex solutions
# Architecture Fulfillment

## Just enough anticipation

<table>
<thead>
<tr>
<th>FF_RM</th>
<th>Architecturally significant features, events and stories are anticipated.</th>
<th>Architects/teams work with stakeholders to identify future events that impact the risk, cost and value of solutions/systems/landscapes</th>
<th>Urgent events that have significant economic impact take architects/teams or stakeholders by surprise.</th>
</tr>
</thead>
</table>

Dependency/critical path analysis is used to start work on the architecture runway in time to mitigate the risk of disrupting events. The timing of realization of enablers is a conscious decision, based on economic trade-offs. Architecture runway improvements are identified late. Teams have to break promises or do significant extra work to prevent disasters. Enablers are only implemented when they become really urgent.
Maturity assessment questions

• Where is your team on the Waterfall Wasteland ↔ Agile Outback map?
• What are the most urgent improvement points to increase your maturity?
Rate this session

Cyberconflict: A new era of war, sabotage, and fear

9:55am-10:10am Wednesday, March 27, 2019
Location: Ballroom
Secondary topics: Security and Privacy

We’re living in a new era of constant sabotage, misinformation, and fear, in which everyone is a target, and you’re often the collateral damage in a growing conflict among states. From crippling infrastructure to sowing discord and doubt, cyber is now the weapon of choice for democracies, dictators, and terrorists.

David Sanger explains how the rise of cyberweapons has transformed geopolitics like nothing since the invention of the atomic bomb. Moving from the White House Situation Room to the dens of Chinese, Russian, North Korean, and Iranian hackers to the boardrooms of Silicon Valley, David reveals a world coming face-to-face with the perils of technological revolution—a conflict that the United States helped start when it began using cyberweapons against Iranian nuclear plants and North Korean missile launches. But now we find ourselves in a conflict we’re uncertain how to control, as our adversaries exploit vulnerabilities in our hyperconnected nation and we struggle to figure out how to deter these complex, short-of-war attacks.

David Sanger
The New York Times

David E. Sanger is the national security correspondent for the New York Times as well as a national security and political contributor for CNN and a frequent guest on CBS This Morning, Face the Nation, and many PBS shows.

© 2019 CGI Inc.
Our commitment to you
We approach every engagement with one objective in mind—to help clients succeed.
References


Contact us to continue the conversation.

Eltjo Poort
Vice-President Architecture
eltjo.poort@cgi.com

linkedin.com/in/eltjopoort
@eltjopoort