An Agile ETL Data Development for ERA XML Submission

Dr Paul Wong
Research Office, Australian National University

ERA – A Complex System?

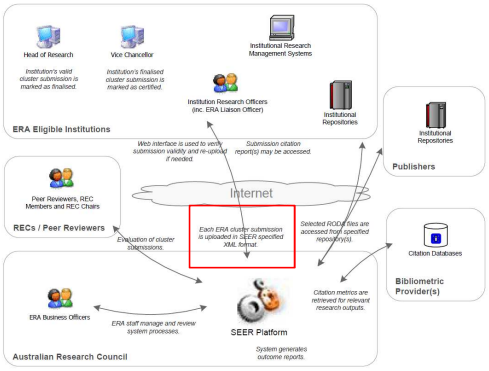
Some characteristics of Complex Systems

- Many actors / stakeholders
- Actors / stakeholders form a network (of interactions)
- Interactions produce emergent behaviours
- Nonlinearity – effects disproportional to (small) changes
- Uncertainty and unpredictability
- Feedback loops – blurring of cause and effect

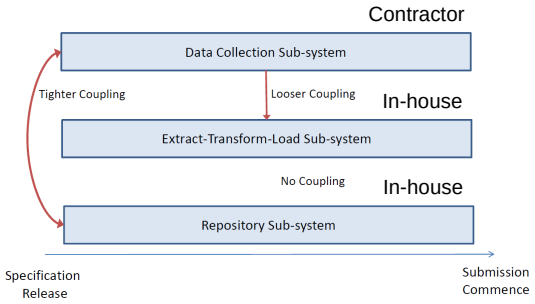
IT / Data Solution in a complex environment

- Technically sound
- Meet the operational constraints
  - Quality (Good)
  - Time to delivery (Quick)
  - Resource (Cheap)
- Cope with the complexity (for us, uncertainty and change)
  - Agility

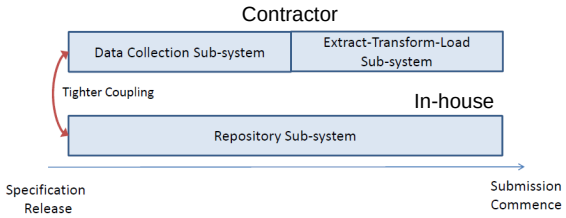
SEER System Interactions



Triple Concurrent Development



Semi-Concurrent Development



The Technical Challenge

- XML
- Validation against XML schema
- Enterprise data stored in relational data structure
- Transformation from relational to XML structure

***Proposition**: It is impossible to implement a fully automatic (and correct) algorithm to transform a given set of data to an xml file (valid) in an arbitrary but given xml schema.*

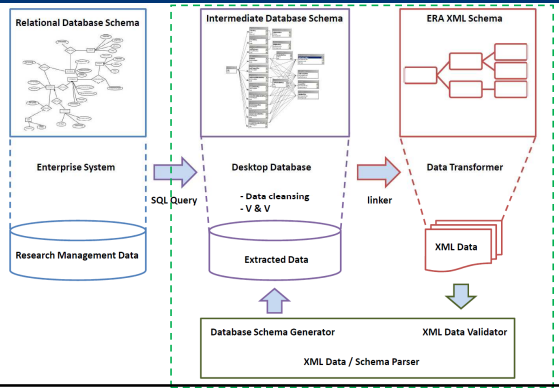
Implications

- Cannot program once and reuse the same program for every xml schema of the eight discipline clusters
- Depending on specification overlap, in the worst case 8 distinct ETL systems will be required
- Hard to estimate time and cost for programming without information of the actual schemata
- Domain knowledge (and hence human intervention) is necessary to map data across different schemata.

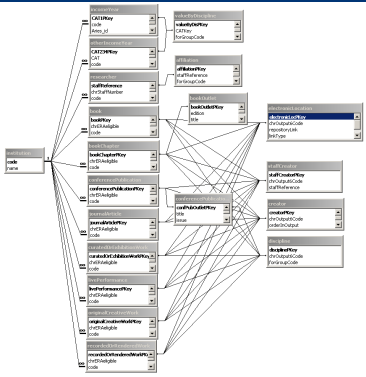
The Technical Solution

- Commercial-Off-The-Shelf (COTS) system architecture
  - A blue print of how components fit together to form the whole system to deliver the required functions
- Build methodology
  - A systematic method to build the underlying system
- Skill, knowledge and understanding
  - Technical know-how
  - Interpret business rules, government guidelines, internal organisation processes and policies

ETL System Architecture

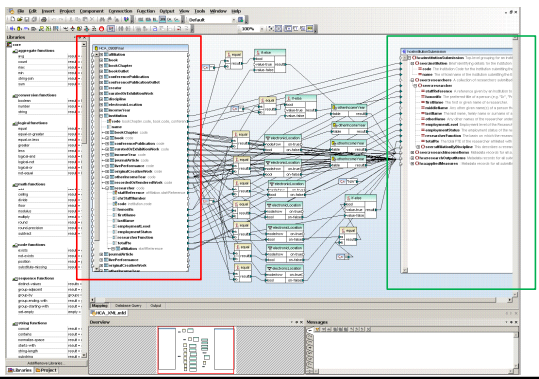


Intermediate Database Schema



- Intermediate database schema is "isomorphic" to the target XML schema
- most work is done in the intermediate database
- no high power programming is required, standard SQL will do
- detailed knowledge of the enterprise data is required

Data Transformer



Main Features of the Architecture

- Plug into the database backend of any Research Management Information System
- Two-pass process of data transformation
- First pass takes a snap shot of enterprise data
- Provide an intermediate step for data cleansing, implementing V&V rules and data debugging
- Intermediate database schema is “isomorphic” to the final target XML schema – simplify data mapping downstream
- No high power programming required, SQL will do

To Cut a Long Story Short

We have

- a COTS system architecture
- a Build methodology
  - A systematic method to build the underlying system
- relevant knowledge
  - technical know-how
  - interpret business rules, government guidelines, internal organisation processes and policies

From Technical to Operational Challenges

We need

- Good – data quality must pass the final V&V in SEER
- Quick – must be delivered on time to SEER
- Cheap – must be within the limit of available resources

In addition

- embrace complexities, changes and uncertainties – must become more agile.

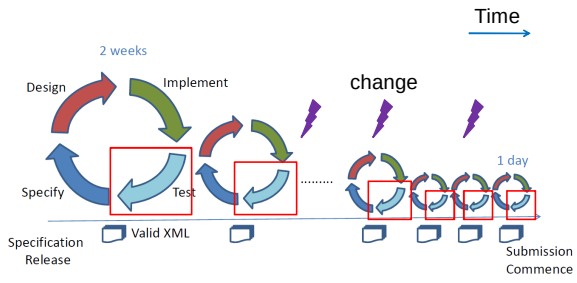
Agile Software / System Development

- New breed of software / system development methodology – offspring of rapid prototyping
- Short iterative development cycles with lightweight processes
- Incremental – add functionalities as we go
- Tacit knowledge – instead of documentation
- Self-organising – collective decision
- Emergence – work structure to be realised during the project instead of a predefined plan

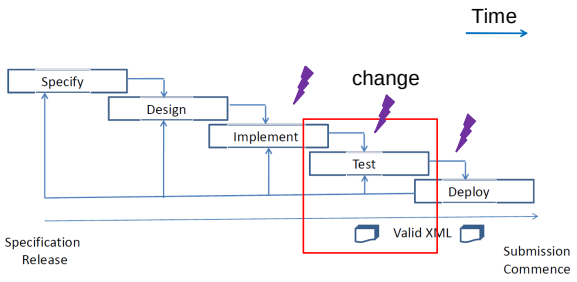
Plan-driven Development

- Traditional approach based on standard engineering practice
- Development moves through a sequence of predefined and standardised processes
- Development phases cannot be easily altered, e.g. no implementation before specification
- Requires completeness of documentation at each phase of the development
- Requires “sign off” at each phase of the development
- Roles, responsibilities and work structure are predefined, monitored and controlled accordingly.

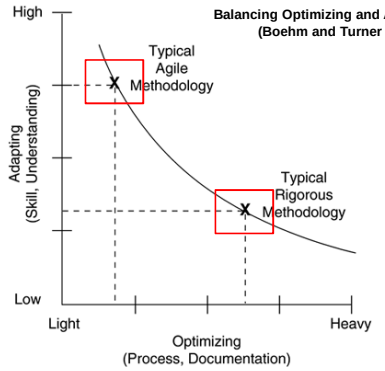
Agile Development



Plan-Driven Development



The People Factor



Cockburn's Levels and Skills (2002)

Level	Characteristics	Plan-driven	Agile
3	Able to revise a method (break its rules) to fit an unprecedented new situation	50% early, 10% throughout	30 % or more
2	Able to tailor a method to fit a precedented new situation	n %	
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience, can become Level 2.	m %	remaining
1B	With training, able to perform procedural method steps (e.g., coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience, can master some Level 1A skills.	30% acceptable	N/A
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.	N/A	N/A

Culture

Plan-driven	Agile
Clear frameworks, policies and procedures	Many degrees of freedom
Calculated, thrive on order	Spontaneous, thrive on chaos
Top-down, conservative	Bottom-up, revolutionary
Hierarchical	Self-organised

Conclusion

- Technical – sound
- Complexity – the right mix of people and methodology
- Operational
  - Good – pass the gold bar V&V in SEER
  - Quick – submitted on the first day SEER was opened
  - Cheap – ???

Question, Comment, Abuse?

Paul.Wong@anu.edu.au