Very Large Calculation Systems

A specialized solution for the complex needs of advanced knowledge workers

Presented by James Madison

About James Madison: *An information architect with over a decade supporting actuaries using the VLCS design*

- Experience
  - Insurance industry since 1995
  - Actuarial systems since 1999
  - The Hartford since 2001
- VLCS experience
  - Built first VLCS starting in 1999
  - Realized it was a pattern when changing companies
  - Never saw it documented in industry literature
  - Wanted to write something on it since 2003
  - CAS call for papers for data processing in 2009
  - Published VLCS paper in 2010
  - Talking to you in 2011
- Education
  - BS in computer science
  - MS in computer science

Disclaimers

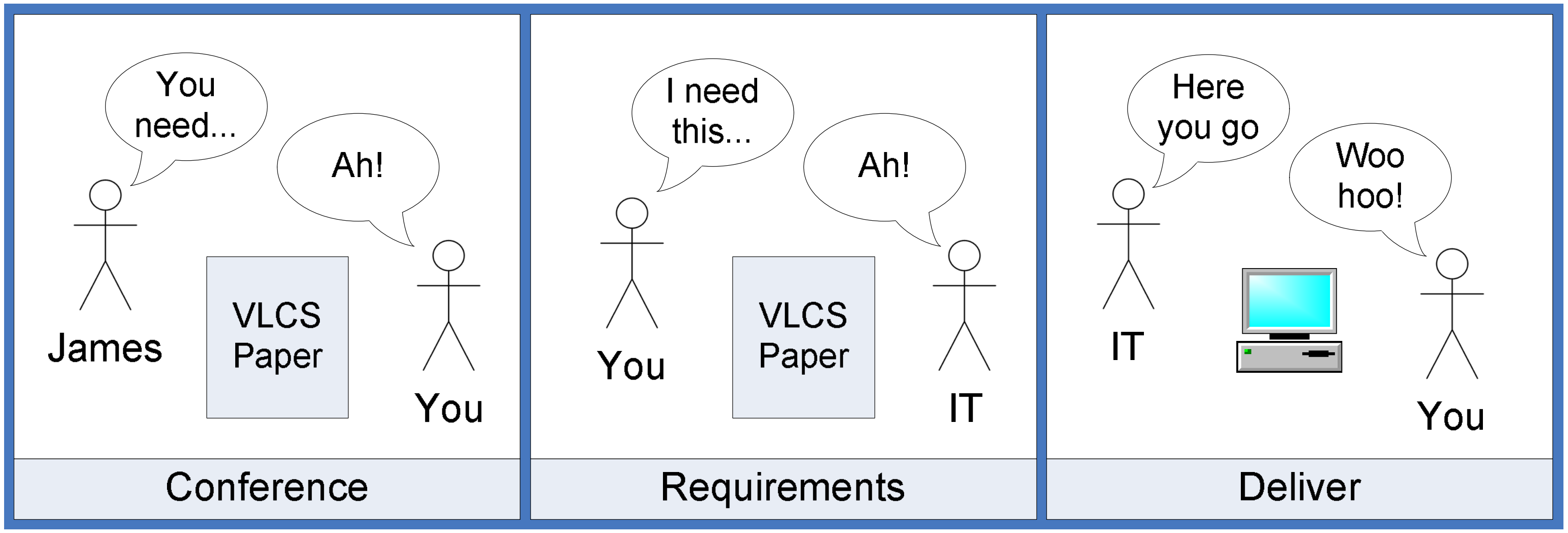
- *Only enough actuarial knowledge to be dangerous*
- *Views not necessarily those of The Hartford or CAS*
- *Vendor/product references are not endorsements*

Objective: *To help you successfully provide specifications for and use a VLCS on the job, should you need one*

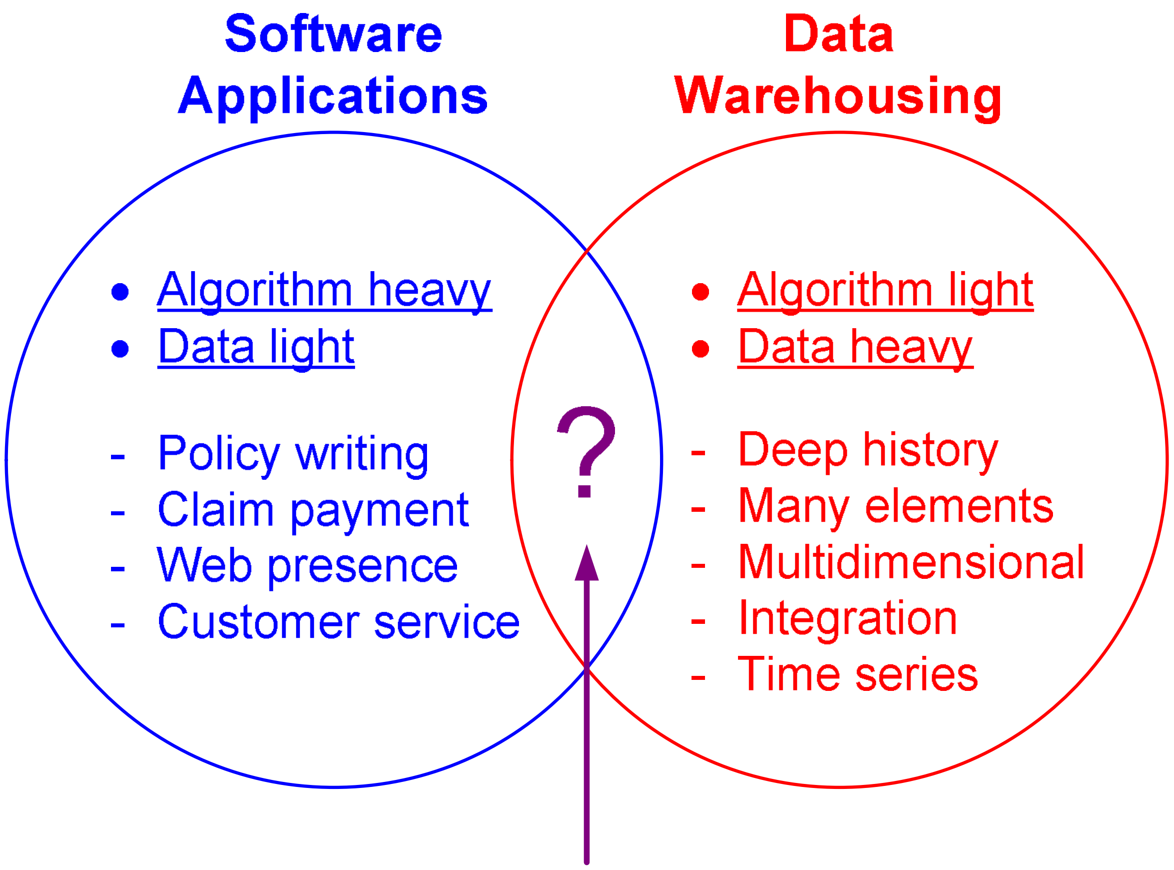
Objective	Summary
Basic design	Large data feeds advanced calculations in flexible environments with high computing power in enterprise systems.
Specific examples	Ratemaking, loss development/reserving, risk analysis. These are just my personal experience. Many others exist.
The alternative	Get strong PCs. Scrounge data. Run spreadsheets. Depend on key people. Hope everyone can find their work in an audit.
When to use it	For large problems whose solution needs a combination of IT power and stability along with flexibility and experimentation.
Value proposition	The combination of computing power and user empowerment is unmatched by any other system design, but it has risks.
How to build one	Deep knowledge of the business domain is the most critical contribution to success.
Technical specifics	Fairly advanced technical elements to know and understand so the IT work can be matched to the need.

Audience: Dual purpose—for actuaries to know what to require in advanced systems, and for IT to know how to deliver it

- For You:
  - Get the basics in this presentation
  - Read paper for more understanding
  - Don’t expect to understand it all
  - Provide to IT with requirements
- For Your IT:
  - Receive paper with requirements
  - Read paper for more understanding
  - Recognize many components
  - Incorporate into actuarial solutions



Basic Design, Motivation: *The pure form of neither software applications nor data warehousing seemed to fit*



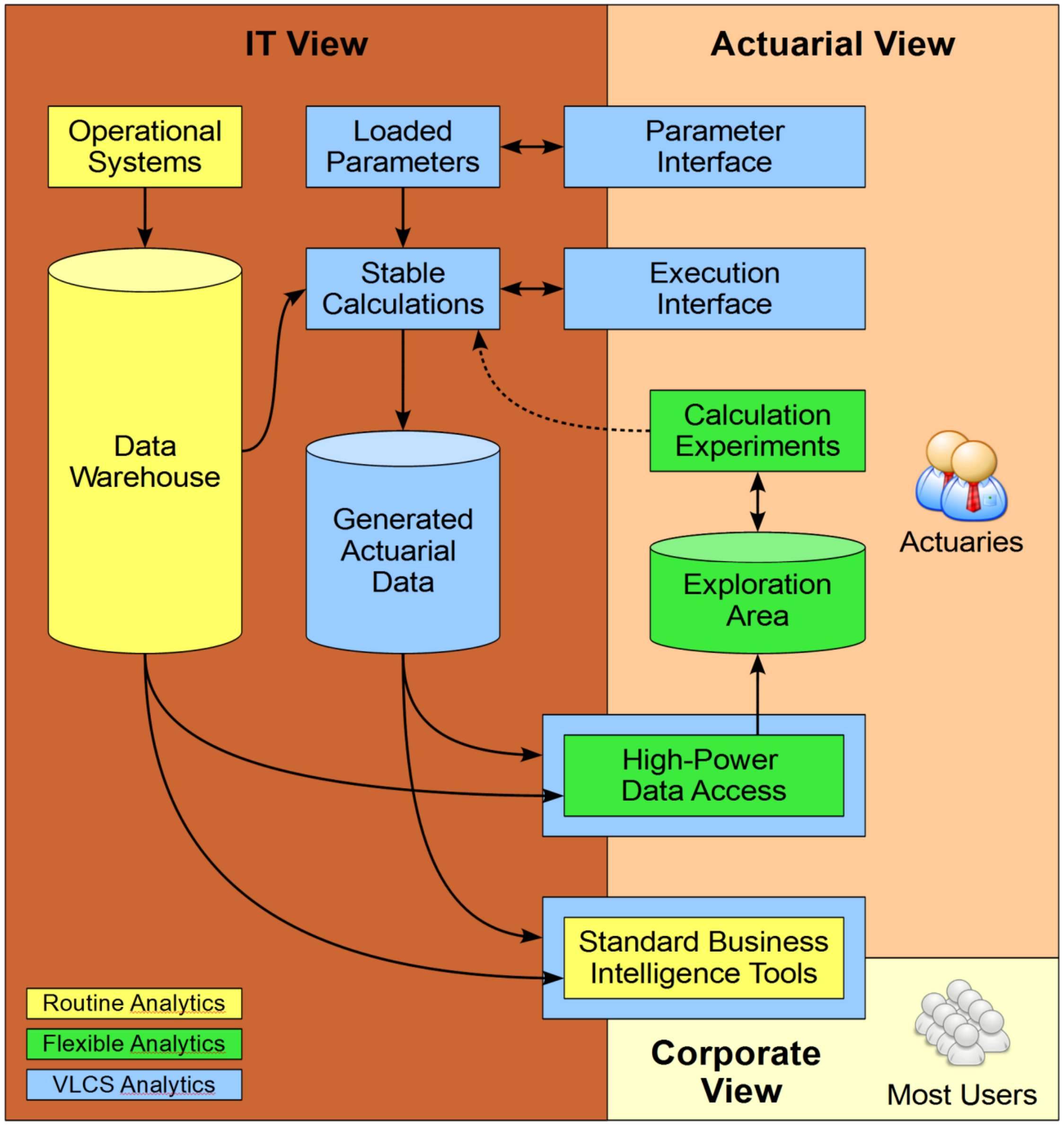
- *Working with actuaries, I kept seeing systems that were not quite applications, not quite data warehousing.*
- *I realized it was a pattern of its own.*
- *Ensure your IT staff know this pattern and have delivered it.*

Very Large Calculation Systems

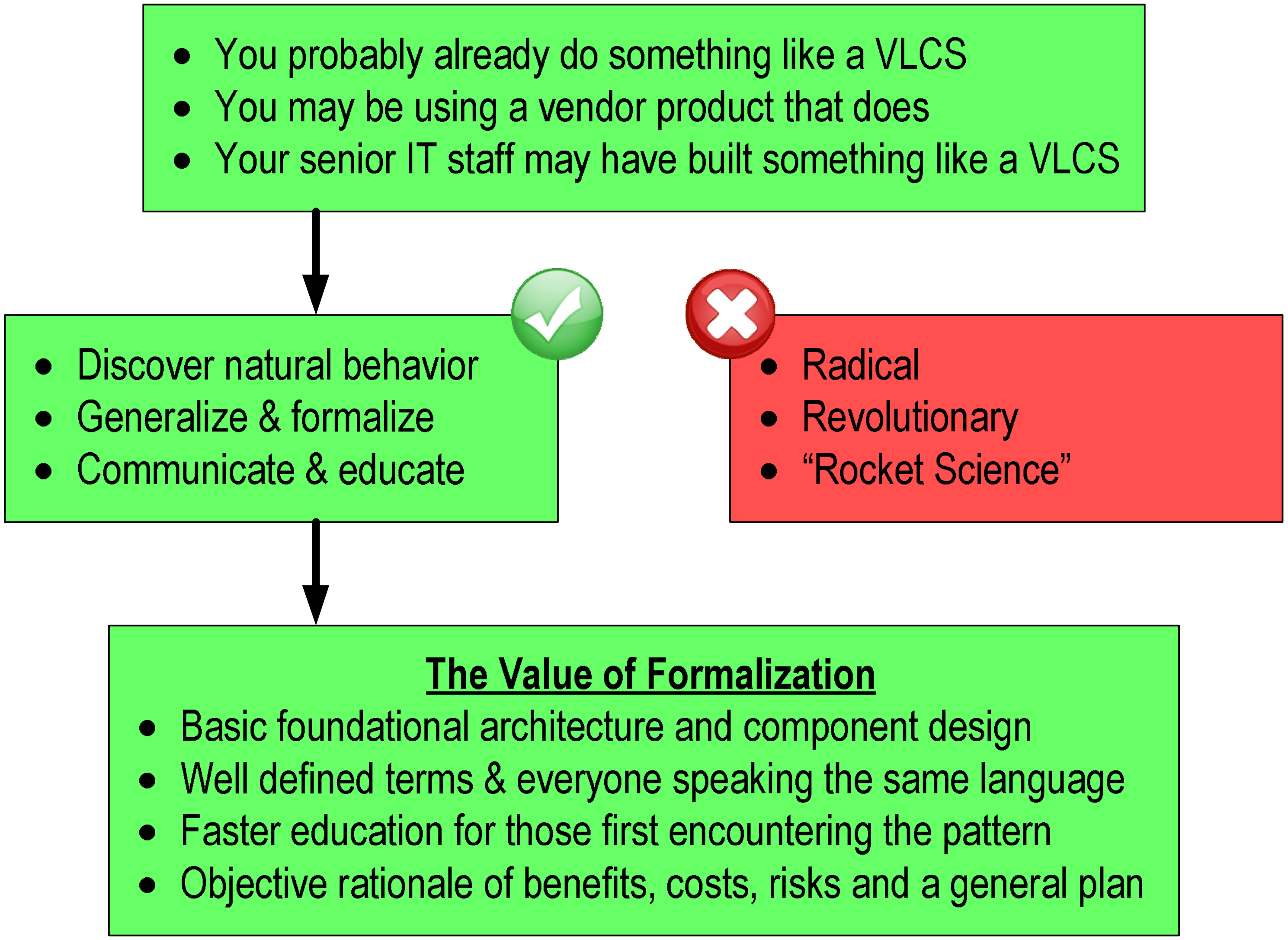
- Algorithm heavy – Ratemaking, loss development, loss reserving, risk
- Data heavy – Many years, many subjects, 3rd party data adds, integrated

Basic Design, Top-Level Architecture: *Data sources, data warehousing, sandboxing, and computing power in a loop*

- Flow order is:
  - Operational Systems
  - Data Warehouse
  - Standard BI Tools
  - High-Power Data Access
  - Exploration Area
  - Calculation Experiments
  - Stable Calculations
  - Loaded Parameters
  - Parameter Interface
  - Execution Interface
  - Generated Actuarial Data
  - *Standard BI Tools*
  - *High-Power Data Access*
  - *(Repeats…)*



Formalization not Revolution: *Most people have done something like this; my hope is to formalize for efficiency*



- You probably already do something like a VLCS
- You may be using a vendor product that does
- Your senior IT staff may have built something like a VLCS

✓

- Discover natural behavior
- Generalize & formalize
- Communicate & educate

✗

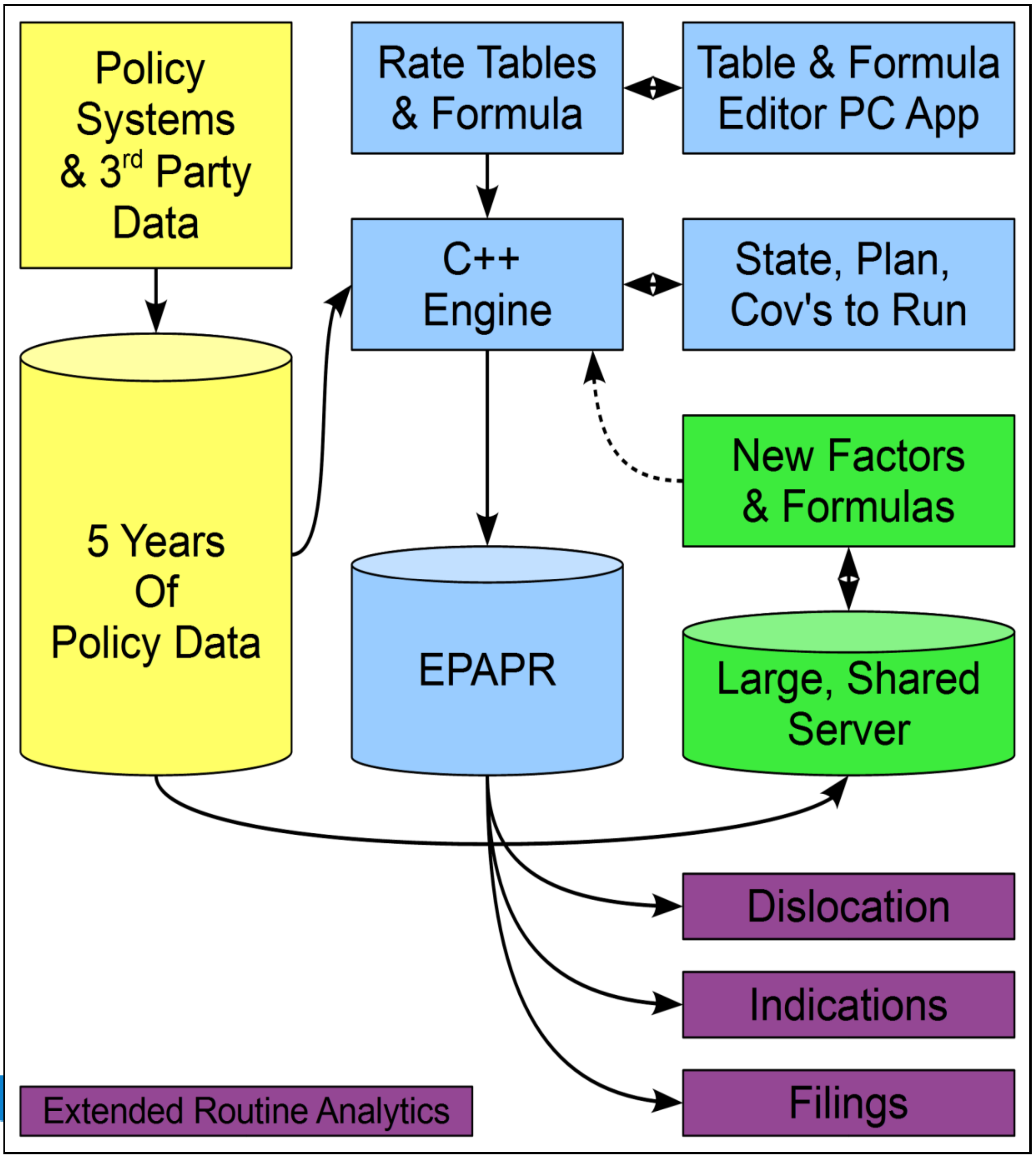
- Radical
- Revolutionary
- “Rocket Science”

<u>The Value of Formalization</u>

- Basic foundational architecture and component design
- Well defined terms & everyone speaking the same language
- Faster education for those first encountering the pattern
- Objective rationale of benefits, costs, risks and a general plan

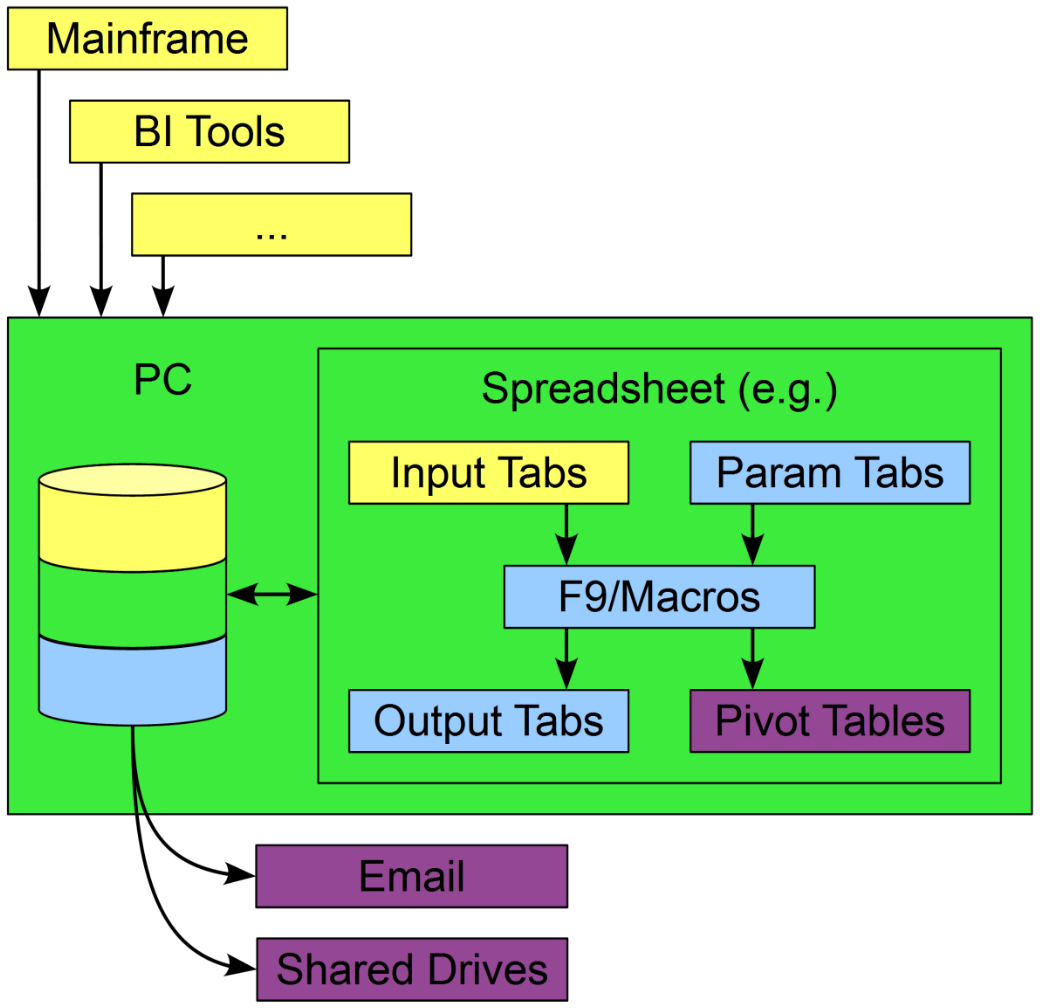
A Specific Example: *Ratemaking for product, pricing, and research teams at enterprise scale with local flexibility*

- Business Goals
  - Enterprise unity
  - Speed to market
  - Both rating & pricing
  - Product support (stable)
  - Research support (dynamic)
  - Product lifecycle in business
- Solution Elements
  - Leading vendor as core
  - Vendor core adaptation
  - Mature data warehouse
  - 3-level sandboxing design
  - Extreme engineering
  - Experienced VLCS team
  - Strong leadership direction



The Alternative: *How to get along without a VLCS; or, how you're already building/using one on your own and don't know it*

- Easiest VLCS I ever built
  - Had run this way for years
  - Then "here, make it a system"
- Value
  - Cheap/easy to start
  - Extreme agility & what-if
- Challenges
  - Hard to share or version
  - Frightening to audit or secure
  - Key person dependencies
  - Weaker algorithms
    - e.g. Parallelogram v. EoE
  - Low computing power
  - Capacity limits
    - e.g. 65K row spreadsheets



When to Use It, Needs: *Watching for the combination of flexibility, stability, and power that indicate the VLCS need*

Criteria	Examples	Rationale
Long History	• 3-5 years for auto • 20+ years for asbestos	• In-force is usually easy • Algorithms across time are hard
Full Book	• Risk classes • Perils by geography • Reaching credibility levels	• Single policy/account is easy • Generalizing insights to reusable/future rules is hard
Complex Algorithms	• Extension of exposures • Loss development • Geographic risk density	• Call center data entry or data warehousing ETL is easy • Time variance, trigonometry, calculus, data mining are hard
Sandbox / What-If	• Experimental ratemaking runs • Testing hypothetical LDFs	• Specifying known rules is easy • Finding new insights is hard
Sufficient Repetition	• Monthly product/pricing review • Real-time risk classification	• Cobble it yourself if it’s rare • Systems repeat reliably & fast

When to Use It, Resources: *Knowing whether you have the basic resources needed to succeed in building a VLCS*

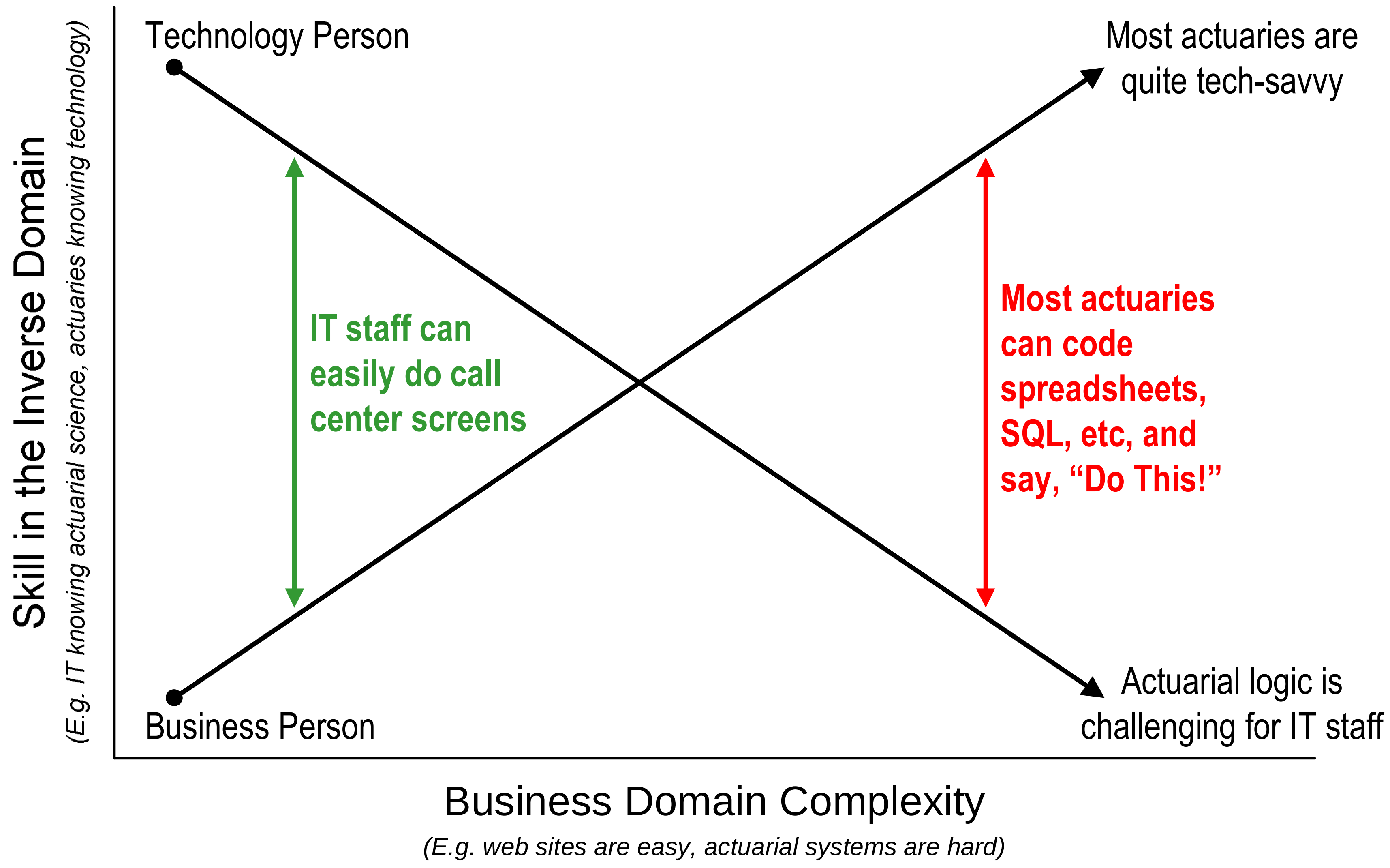
Criteria	Examples	Rationale
Power Users	• Comfortable coding themselves • Many automated tools already	• The stronger the actuary, the smaller the gap to IT building it
A Data Warehouse	• Many source already together • Integration headaches resolved	• Collecting sources and unifying them is very time consuming; do not attempt simultaneously
Hardware Power	• Many multi-core CPUs • Commodity servers as a grid	• Once all else is optimized, raw power will still matter
Project Mgt Skill	• PM who has built a VLCS • Experience with "Agile" SDLC	• Iterative, incremental build with involved business community is needed but often not the norm
Enterprise Will Power	• Sustained multi-year effort • Analytics-aware funding model	• Complete system will require several years to fully construct • A VLCS is a "back room" system, so harder to allocate business benefit

Value Proposition: *The pros and cons of using a VLCS compared to applications, data warehousing, or doing it yourself*

Pro/Con	Application	Data Warehouse	Do-It-Yourself
Flexibility	Same	More	Less
Self-Service	Same	More	Less
History	More	Same	More
Algorithm Power	Same	More	More
Computing Power	More	Same	More
Auditability	Same	Same	More
Formalization	Same	Same	More
Vendor Products	Less	Less	Less
Cost	More	More	More
Risk	More	More	More
Complexity	More	More	More
Manageability	Same	Same	More

Read as: “A VLCS has {cell} {row pro/con} than {column header}”

How to Build One, Perspective: *You know the technology domain better than IT knows the actuarial domain*


Technology Person
Most actuaries are quite tech-savvy
IT staff can easily do call center screens
Most actuaries can code spreadsheets, SQL, etc, and say, “Do This!”
Business Person
Actuarial logic is challenging for IT staff
Skill in the Inverse Domain
(E.g. IT knowing actuarial science, actuaries knowing technology)
Business Domain Complexity
(E.g. web sites are easy, actuarial systems are hard)

How to Build One, Contributions: *The assistance you can provide to ensure that you get the VLCS you need*

Action	Rationale
Code it yourself	Build what you need into spreadsheets or databases, hand over, say "Make it do this!"
Use IT tools	As you code, use sanctioned IT tools if you can. Maximizes knowledge transfer; minimizes cost.
Say *how* not just *what*	Traditional IT asks for the inverse. Spell out how—you will often be providing a big head start.
Clarify flexibility versus stability	Making flexibility systematic is not a common IT skill. Spell out clearly where you need it and where you don't.
Decompose & prioritize	Make units of delivered functionality small and ensure execution in priority order.
Demand "Agile" SDLC	Use iterative, light-weight, collaborative development. Internet search on "agile software development" for specifics.
Ask "how hard is that?"	Not just in a VLCS, but with any software development, this is a powerful way to find confused IT people and help them.
Educate on algorithms	Don't just "do specs." Teach IT actuarial science. E.g. "*Basic Ratemaking*" by CAS—great work!

Technical Specifics: *Moderately advanced technical and design elements to watch for and know the value of*

Feature	Description	Value
Parameters	User guides scope and input of job	Flexibility, what-if analysis
Parallelism	Many jobs run at once	Higher performance
Partitioning	Only needed data is retrieved	Higher performance
Profiling	See where run time is spent; per line	Higher performance
Cluster/grid	Many low-cost servers together	Lower cost
Networking	Server connections are fast	Higher performance
Self-service	Users invoke VLCS on demand	Flexibility, what-if analysis
Job priority	Jobs have classes and order	Enterprise management
Queuing	Maximum job limits are used	Consistent performance
Monitoring	Users can see system load	Consistent performance
Alerting	System notifies user when done	Enterprise management
Archiving	Any system run can be tracked	Auditing & compliance
Sharing	Users can see and reuse others' jobs	Non-redundancy, performance

Summary

- Paper from which this presentation is drawn:
  - http://www.casact.org/pubs/forum/10spforum/Madison.pdf
- Contact information
  - James.Madison (at) TheHartford.com
- Q&A
  - Any questions?