Architecture of Enterprise Applications II
Arch. & Architect

Haopeng Chen

***RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)*
Shanghai Jiao Tong University
Shanghai, China
e-mail: chen-hp@sjtu.edu.cn

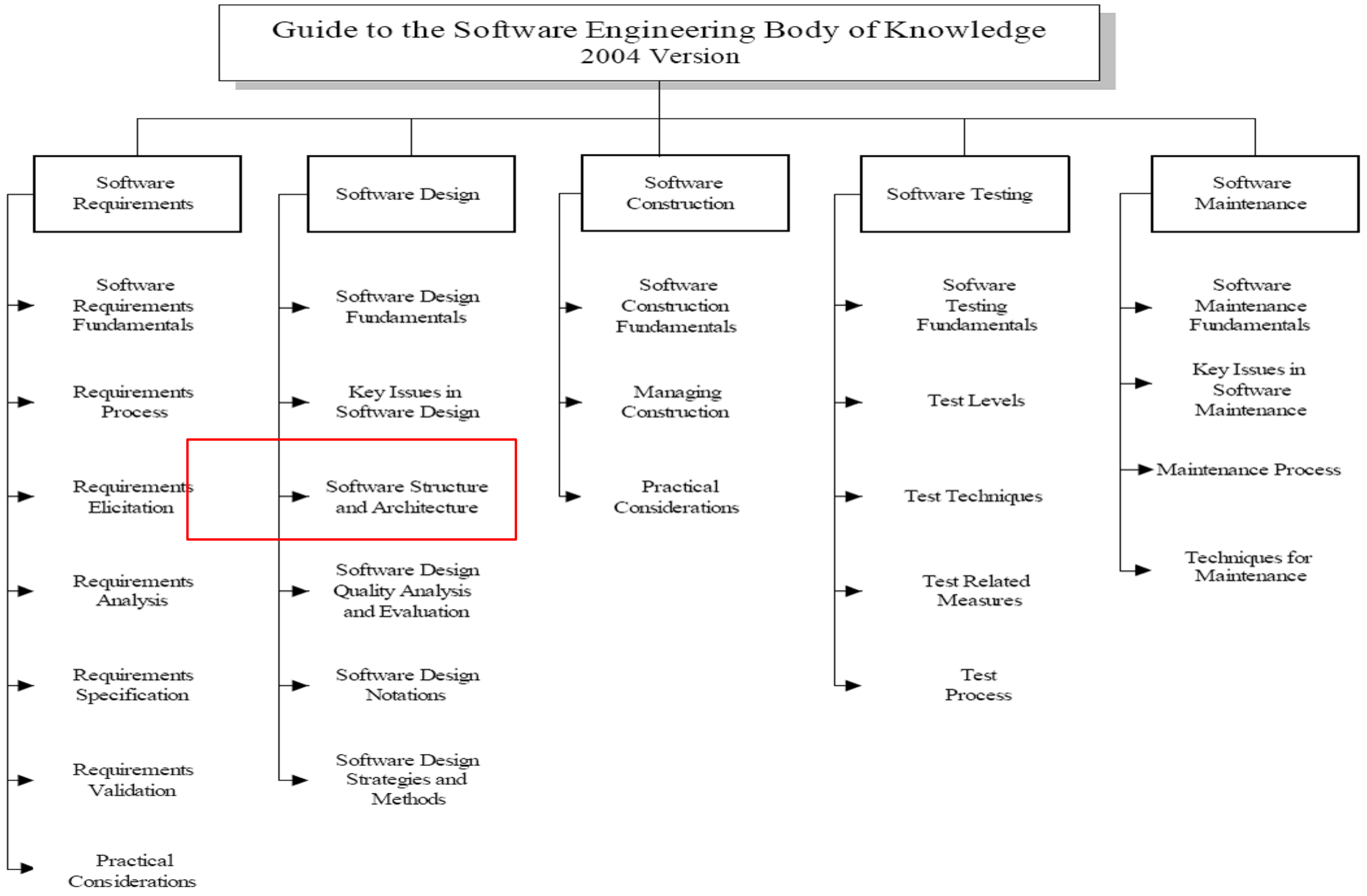
Agenda

- Introduction to architecture of enterprise apps
  - Definition and content of architecture
  - Qualification of architects
  - Features of Arch. Of Enterprise Apps.

SWEBOK

REliable, INtelligent & Scalable Systems

- Guide to the Software Engineering Body of Knowledge
  - A project of the IEEE Computer Society Professional Practices Committee

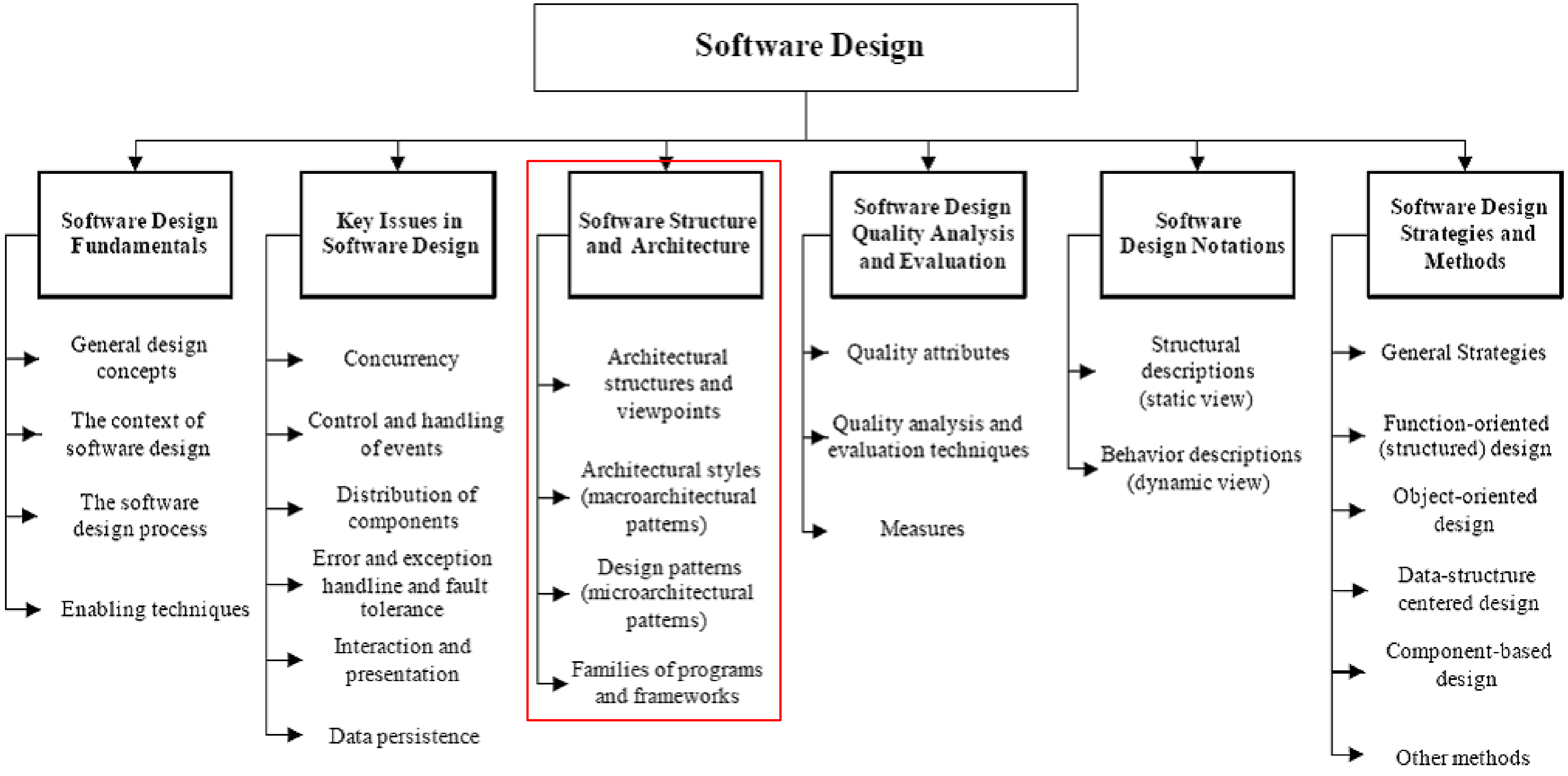


Guide to the Software Engineering Body of Knowledge 2004 Version
Software Requirements
Software Requirements Fundamentals
Requirements Process
Requirements Elicitation
Requirements Analysis
Requirements Specification
Requirements Validation
Practical Considerations
Software Design
Software Design Fundamentals
Key Issues in Software Design
Software Structure and Architecture
Software Design Quality Analysis and Evaluation
Software Design Notations
Software Design Strategies and Methods
Software Construction
Software Construction Fundamentals
Managing Construction
Practical Considerations
Software Testing
Sofware Testing Fundamentals
Test Levels
Test Techniques
Test Related Measures
Test Process
Software Maintenance
Software Maintenance Fundamentals
Key Issues in Software Maintenance
Maintenance Process
Techniques for Maintenance

SWEBOK

REliable, INtelligent & Scalable Systems

- Guide to the Software Engineering Body of Knowledge
  - A project of the IEEE Computer Society Professional Practices Committee

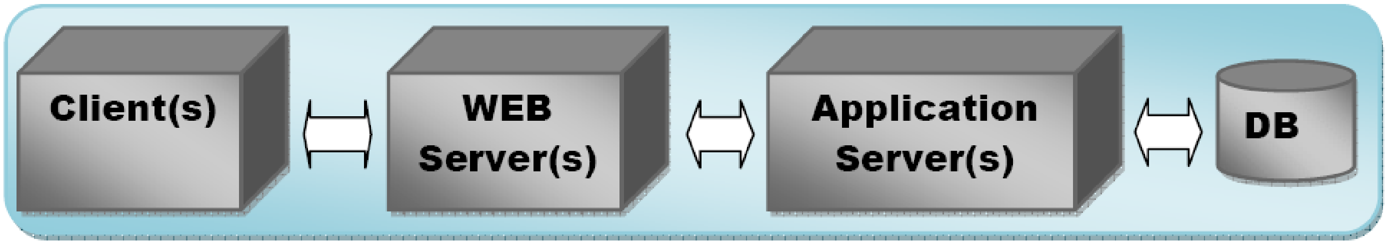


Software Design
Software Design Fundamentals
General design concepts
The context of software design
The software design process
Enabling techniques
Key Issues in Software Design
Concurrency
Control and handling of events
Distribution of components
Error and exception handline and fault tolerance
Interaction and presentation
Data persistence
Software Structure and Architecture
Architectural structures and viewpoints
Architectural styles (macroarchitectural patterns)
Design patterns (microarchitectural patterns)
Families of programs and frameworks
Software Design Quality Analysis and Evaluation
Quality attributes
Quality analysis and evaluation techniques
Measures
Software Design Notations
Structural descriptions (static view)
Behavior descriptions (dynamic view)
Software Design Strategies and Methods
General Strategies
Function-oriented (structured) design
Object-oriented design
Data-structrure centered design
Component-based design
Other methods

RUBiS

REliable, INtelligent & Scalable Systems

- RUBiS http://rubis.ow2.org/
- How is RUBiS built?



What is Software Architecture ?

- "Architecture" is a term that lots of people try to define, with little agreement.
- There are two common elements:
  - One is the highest-level breakdown of a system into its parts;
  - the other, decisions that are hard to change.
- It's also increasingly realized that there isn't just one way to state a system's architecture; rather, there are multiple architectures in a system, and the view of what is architecturally significant is one that can change over a system's lifetime.

Why Software Architecture?

- Increasing Size and Complexity of Software Systems.
- Designing and Specifying the overall system structure.
- Gross organization and global control of systems structure.
- Scaling and Performance.
- Selection among design alternatives.

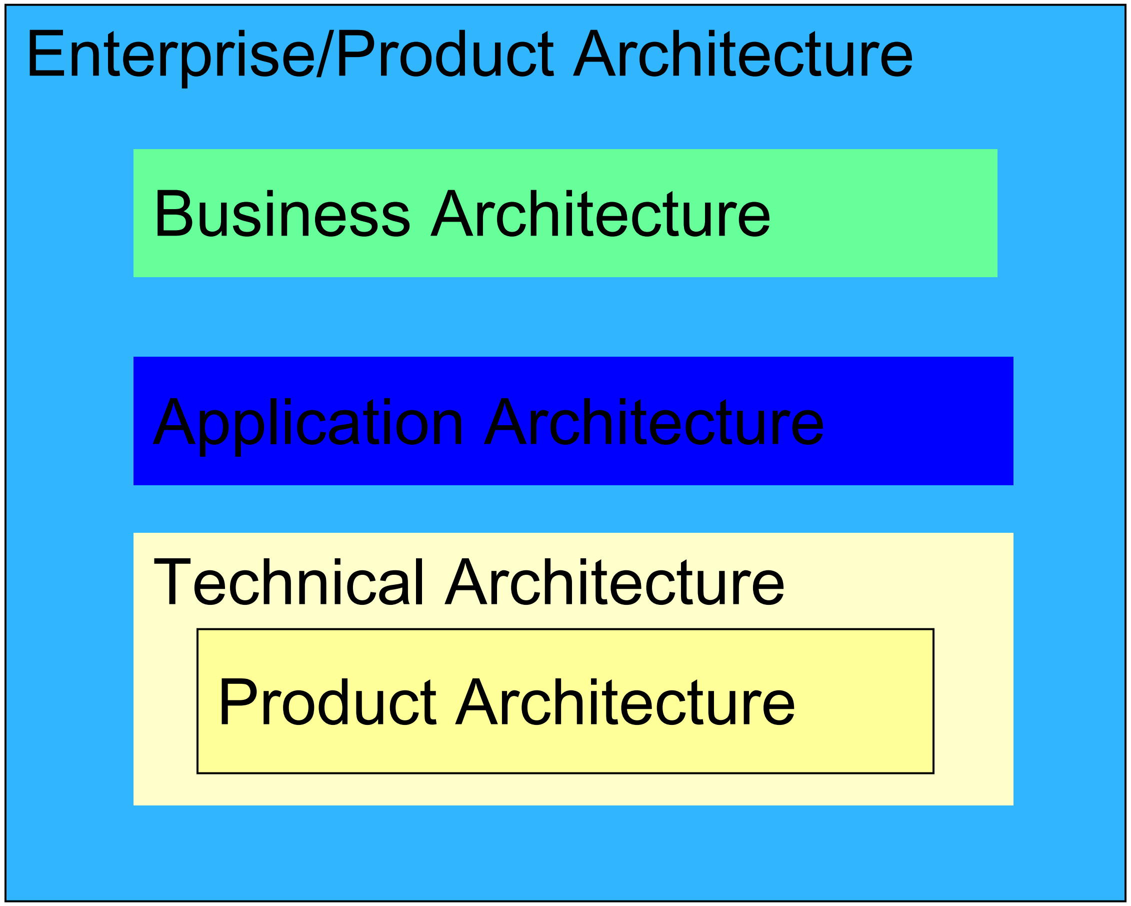
Software Architecture in SWEBOK

- In its strict sense, a *software architecture is "a description* of the subsystems and components of a software system and the relationships between them".

- Architecture thus attempts to define the internal *structure*—according to the *Oxford English Dictionary, "the way in* which something is constructed or organized"—of the resulting software.

What is Software Architecture

Typical System Architecture



What is Software Architecture

- Software Architecture transforms Business Architecture into an set of designs and guidelines to realize business process in an information systems

- Software Architecture relies on Technical Architecture to provide an efficient/scalable/secure environment

Software Architecture-SWEBOK

- *Architectural Structures and Views*
  - Different high-level facets of a software design can and should be described and documented. These facets are often called views: “A view represents a partial aspect of a software architecture that shows specific properties of a software system”
    - logical view (satisfying the functional requirements)
    - process view (concurrency issues)
    - physical view (distribution issues)
    - Development view (how the design is broken down into implementation units).
  - a software design is a multi-faceted artifact produced by the design process and generally composed of relatively independent and orthogonal views.

Software Architecture-SWEBOK

- *Architectural Styles*
  - An architectural style is "a set of constraints on an architecture that defines a set or family of architectures that satisfies them"
  - An architectural style can thus be seen as a meta-model which can provide software's highlevel organization (its macroarchitecture).
    - General structure (for example, layers, pipes and filters, blackboard)
    - Distributed systems (for example, client-server, threetiers, broker)
    - Interactive systems (for example, Model-View-Controller, Presentation-Abstraction-Control)
    - Adaptable systems (for example, micro-kernel, reflection)
    - Others (for example, batch, interpreters, process control,
    - rule-based).

Software Architecture-SWEBOK

- *Software Architecture for Product Families*
  - One possible approach to allow the reuse of software designs and components is to design families of software, also known as software product lines. This can be done by identifying the commonalities among members of such families and by using reusable and customizable components to account for the variability among family members.
  - In OO programming, a key related notion is that of the framework: a partially complete software subsystem that can be extended by appropriately instantiating specific plug-ins (also known as hot spots).

Components of Software Architecture

- Business model
- Platform
- Layer/Network Model
- Domain Framework
- Technical Framework
- Deployment Model

What HW/SW components are involved

REliable, INtelligent & Scalable Systems

- Database
- Application Server/Middleware
- Network
- Client Devices
- Servers
- Other products

Components of Technical Architecture

- Hardware requirements (Server) for Database
- Hardware requirements (Server) for Application Server, Middleware
- Hardware requirements for Client Devices
- Network design for Data center where Servers are hosted and connection among client devices, offices to data center
- Security HW/SW design in the whole systems

Components of Application Architecture

- Domain Models
- Programming model
- Platform Architecture
- Implementation Architecture
- Components
- Development Environment
- Development guidelines

Software Architecture Quality Metrics

- Availability
- Reliability
- Modifiability
- Performance
- Security
- Testability
- Usability
- Supportability

Software Architecture Quality Metrics

- Business objectives(T2M,Targeted Market)
- Skill of development team and local market of team
- Cost to build and Maintain v.s. Benefit
- Materiality of Technology
- Current System constrain, integration
- Migration, migration, migration

Summary

- Software Architecture need to consider from both technical point of view as well as business point of view
- There are different ways to communicate architecture design
- There is no THE best architecture for any one software system
- Always need consider
  - external constrain, such cost, infrastructure,…
  - Maturity of IT organization, not only development, but also operations
- Architecture is a live entity, therefore, ability to grow, migrate is very important

What is Software Architect

- Circa 25 BCE, Vitruvius described the role of an architect as:
  - *The ideal architect should be a man of letters,*

    a mathematician, familiar with historical studies,

    a diligent of philosophy, acquainted with music,

    not ignorant of medicine, learned in the responses of Jurisconsultis, familiar with astronomy and astronomical calculations

Risk and Rewards of Architect

- Risks
  - you don't enjoy the non-technical work
  - more responsibility / less control
  - insufficient drive to overcome resistance
  - poor odds of success
  - everyone has a better idea
- Rewards
  - more interesting and complex problems
  - career advancement and recognition
  - greater scope of activities, influence, contribution

What Architect Do

- Gather System Requirements
- Translate requirement to architecture design
- Implement proof of concept, define development environment
- Develop Architecture Module
- Communicate Architecture Design
- Be the technical expert
- Mentor/help Developers

Architect need to be ---

Architect needs to be good at

- seeing the big-picture, abstracting
- dealing with ambiguity
- setting priorities
- dealing with conflicting priorities, making compromises
- analyzing tradeoffs
- working across disciplines
- leading, following and getting out of the way
- persuading others
- mediating conflicts

Technical Skills

REliable, INtelligent & Scalable Systems

- Requirement gathering/Management
- Modeling and analysis methodology
- Full Software Development Life Cycle
- Modern architectural technologies, such as J2EE and .NET
- In depth knowledge of programming languages
- Network, Security, hardware platforms
- Database

None Technical Skills

- Facilitation/consulting
- Communication/Presentation/Sales skill
- Mentoring
- Domain Knowledge of area you working on
- Leadership
- Organizational politics
- Business acumen/Strategy

What is a the best Architect

REliable, INtelligent & Scalable Systems

- The best architects are good technologists and command respect in the technical community, but also are good strategists, organizational politicians (in the best sense of the word), consultants and leaders

Things to remember

1. Focus on people, not technology or techniques. The greatest architecture model in the world has little value if you can't find a way to work with developers effectively and convince them to use it.

2. Keep it simple. The simpler the architecture, the greater the chance that it will be understood and actually followed by developers.

Things to remember

3. Work iteratively and incrementally. Your architecture will evolve over time due to new requirements, new technological choices, and greater understanding amongst your enterprise architecture team.

4. Roll up your sleeves. Developers won't respect you, and therefore won't accept your architecture, if you aren't willing to get actively involved in their project efforts.

Things to remember

REliable, INtelligent & Scalable Systems

5. Build it before you talk about it. Everything works in diagrams but can fail miserably in practice. Just like in the RUP, you should prove your application architecture via technical prototyping; there is no reason why you can't do the same at the enterprise level.

6. Look at the whole picture. This is a primary skill of enterprise architects, which is one of the reasons why a multi-view approach is so important to you

Things to remember

7. Make architecture attractive to your customers. If your enterprise architecture artifacts aren't easy to understand, to access, and to work with, your customers (developers and senior managers) will very likely ignore your work.

Conclusion

- How do I start
  - Become an excellent developer who knows why not just how
  - Understand the relationship of underline technical architecture and software systems
  - Understand current platform specific architecture, J2EE and .NET
  - Know the domain you are working on, become an expert
  - Read, read, read, think think think

What is Enterprise Application

- Enterprise applications often have complex data -- and lots of it -- to work on, together with business rules that fail all tests of logical reasoning.
  - Other terms for enterprise applications include “information systems"
  - Enterprise applications include payroll, patient records, shipping tracking, cost analysis, credit scoring, insurance, supply chain, accounting, customer service, and foreign exchange trading.
  - Enterprise applications don't include automobile fuel injection, word processors, elevator controllers, chemical plant controllers, telephone switches, operating systems, compilers, and games.

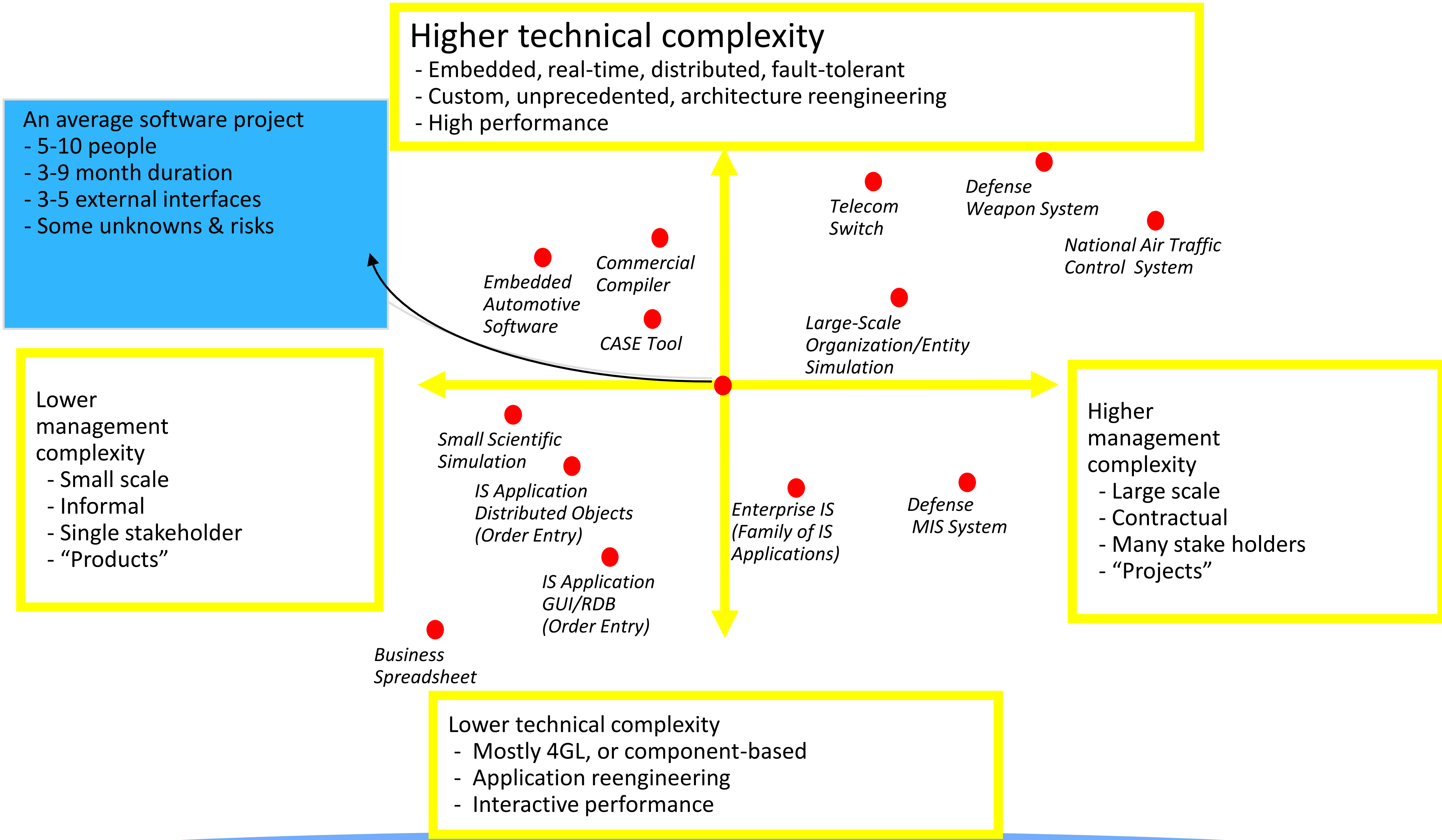
What is Enterprise Application

- Enterprise applications
  - usually involve persistent data
  - usually have a lot of data
  - usually many people access data concurrently
  - usually have a lot of user interface screens
  - usually they need to integrate with other enterprise applications scattered around the enterprise
  - conceptual dissonance with the data
  - complex business "illogic"

Enterprise Applications

- An enterprise system is one that has the following qualities:
  - Shares some or all of the resources used by the application
  - Is intended for internal use
  - Must work within existing architecture
  - Will be deployed and supported by internal IT staff
  - Requires greater robustness, both in terms of exception-handling and scalability
  - Must fail gracefully
  - Must gracefully handle evolution over time

Dimensions of software complexity


Higher technical complexity
- Embedded, real-time, distributed, fault-tolerant
- Custom, unprecedented, architecture reengineering
- High performance
An average software project
- 5-10 people
- 3-9 month duration
- 3-5 external interfaces
- Some unknowns & risks
Telecom Switch
Defense Weapon System
National Air Traffic Control System
Commercial Compiler
Embedded Automotive Software
CASE Tool
Large-Scale Organization/Entity Simulation
Lower management complexity
- Small scale
- Informal
- Single stakeholder
- "Products"
Higher management complexity
- Large scale
- Contractual
- Many stake holders
- "Projects"
Small Scientific Simulation
IS Application Distributed Objects (Order Entry)
Enterprise IS (Family of IS Applications)
Defense MIS System
IS Application GUI/RDB (Order Entry)
Business Spreadsheet
Lower technical complexity
- Mostly 4GL, or component-based
- Application reengineering
- Interactive performance

REin

Thank You!