What characteristics are right?

- **Need to be able to relax the strict ACID properties**
- **Need to put control of some into hands of service developer**
  - Is consistency (or consensus) important?
- **May need a notion of a central coordinator**
  - But may not!
  - Or something with a fuzzy idea of what's going on
- ***"A comparison of Web services transaction protocols"*, IBM Developer Works, 2003.**

Relaxing isolation

- **Internal isolation or resources should be a decision for the service provider**
  - E.g., commit early and define compensation activities
  - However, it does impact applications
    - Some users may want to know a priori what isolation policies are used
- **Undo can be whatever is required**
  - Before and after image
  - Entirely new business processes

Relaxing atomicity

- **Sometimes it may be desirable to cancel some work without affecting the remainder**
  - E.g., prefer to get airline seat now even without travel insurance
- **Similar to nested transactions**
  - Work performed within scope of a nested transaction is provisional
  - Failure does not affect enclosing transaction
- **However, nested transactions may be too restrictive**

Structuring transactions

- **Could structure transactional applications from short-duration transactions**
  - Release locks early
  - Resulting application may still be required to appear to have “ACID” properties
    - May require application specific means to restore consistency
- **A transactional workflow system could be used to script the composition of these transactions**

Relaxation of consistency

- **ACID transactions (with two-phase commit) are all about strong global consistency**
  - All participants remain in lock-step
  - Same view of transaction outcome (atomic)
- **But that does not scale**
  - Replication researchers have known about this for years
    - Weak consistency replication protocols developed for large scale (number of replicas and physical deployment)
    - Merging of caching and replication protocols
      - Local domains of consistency
    - Cannot “stop the world” and enforce global consistency
  - Some transaction research into this, but industry pushing global consistency
    - Starting to see a change

Heisenberg's Uncertainty Principle

- **Cannot accurately measure both position and momentum of sub-atomic particles**
  - Can know one with certainty, but not the other
  - Non-deterministic measurements
- **Large-scale/loosely-coupled transactional applications suffer the same effect**
  - Can know that all services will eventually see same state, just not when
  - Or at known time can determine state within model/application specific degree of uncertainty
- **Or another way of thinking about it …**
  - No such thing as simultaneity in data space as there isn't in space-time
    - *"Data on the Outside vs. Data on the Inside", by Pat Helland*

No global consensus

- **Split transactions into domains of consistency**
  - Strong consistency within domains
  - Some level of (known) consistency between domains
    - See *"A method for combining replication and caching", Proceedings of International Workshop on Reliable Middleware Systems, October 1999*.
    - *OASIS WS-BusinessProcess specification*, part of OASIS WS-CAF, 2003.
  - Resolve inconsistencies at the business level
    - Don't try and run consensus protocols between domains
- **Consistency related to isolation**
  - Put into the control of the service and application developers

OASIS Business Process

- **All parties reside within *business domains***
  - Recursive structure is allowed
  - May represent a different transaction model
  - No required notion of consistency between domains
- **Business process is split into *business tasks***
  - Execute within domains
  - Compensatable units of work
    - Forward compensation during activity is allowed
      - Keep business process making forward progress
- **Consistency is application (service) dependent**
- **Atomicity (or lack thereof) in the “large” is taken for granted**

SOA or scale?

- **Problems with transactions pre-date SOA**
- **Current issues with database technologies are not SOA specific either**
- **Problems are two-fold**
  - Scalability (size and geographic distributed nature)
  - Control over the infrastructure/services
    - Trust comes into this too
- **Much research in the 1990's**
- **SOA (and Web Services) bring this to the foreground**
  - REST would be just as appropriate

Future directions

- **One size does not fit all!**
- **Business domains will impose different requirements on implementers**
  - Essentially construct domain-specific models
  - Real-time
- **The range and requirements for such extended models are not yet known**
  - Do not restrict implementations because we don't know what we want yet
- **Still a very active area of research and development**