Transactions and SOA

Tuukka Ruotsalo

Contents

- Transactions, what are they, why do we have to think about transactions in SOA?
- Transactional requirements
  - ACID
- Distributed & long living transactions
  - Atomic commitment protocol
    - Two-phase commit & cooperative termination
- Nested transactions
  - Compensation & Saga transactions
- Forward recovery

Transactions

- RPCs support a single-call-at-time interactions between clients and services
- As soon as we go beyond simple single service invocations we need to make sure that the data is not corrupted in any sense
  - Most of the SOA solutions are more complicated than single calls
- WS specification support
  - WS-Coordination
  - WS-Atomic Transactions
  - WS-Business Activity

Example (Newcomer, 2002)

- Skate boots ordering
  - Authentication
  - Credit validation
  - Accounting
  - Probably more:
    - Informing the buyer that the boots are on their way
    - Confirming that the goods have arrived

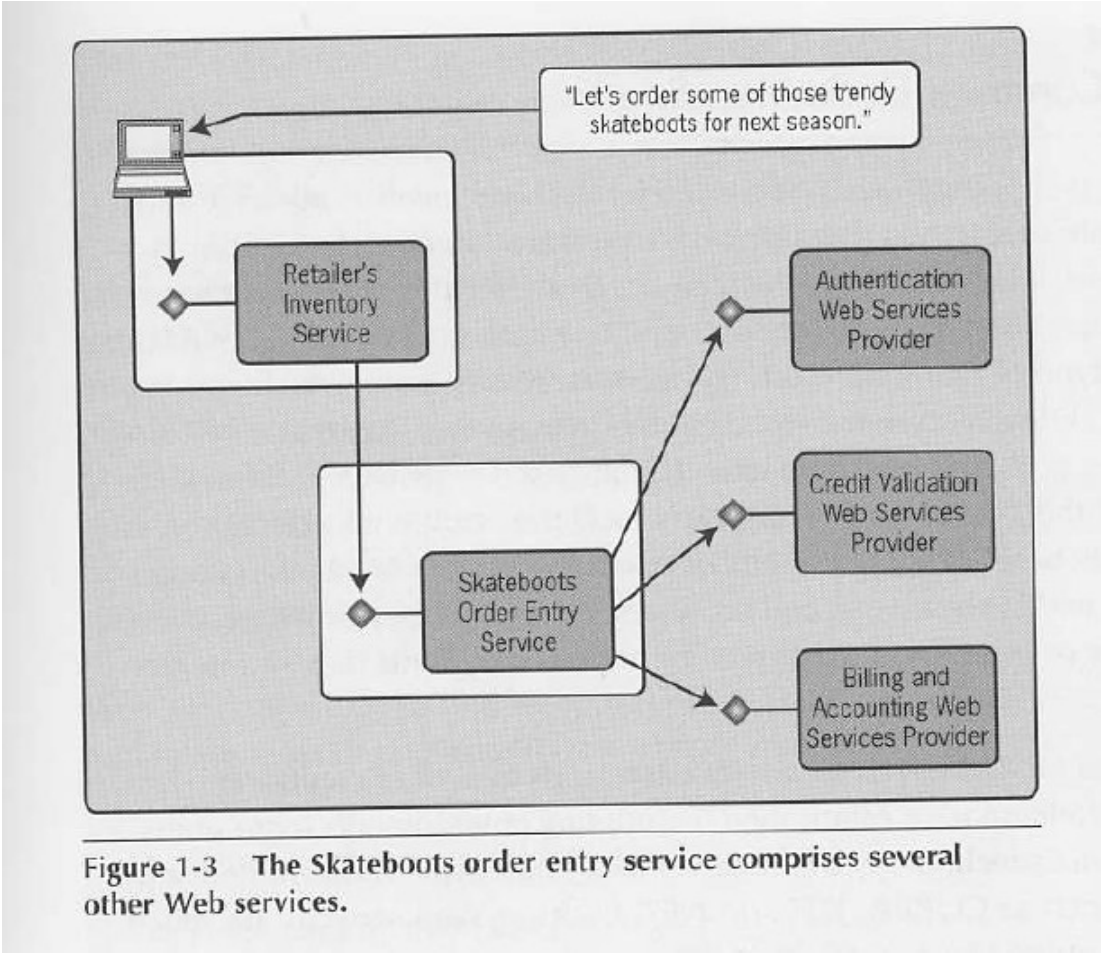


Figure 1-3 The Skateboots order entry service comprises several other Web services.

Transactions ACID

- ACID defines requirements for transactions
- Designed for short-lived decisions usually lasting under a few seconds.
- Identify a logical unit of work that is either performed in complete or not at all. That is, either a COMMIT or a ROLLBACK is performed on the operations. This enables the data to maintain a state of consistency.
- ACID
  - Atomicity
  - Consistency
  - Isolation
  - Durability

Transactions ACID

- Atomicity
  - The actions that transaction performs in the system are committed only if all of the actions are successful, otherwise roll-back
  - If an error occurs the original state of the system is recovered.

Transactions ACID

- Consistency
  - A consistent state of the system may be expected at all times
  - Changes in the system must correspond to the actual changes and they must be done by the integrity constraints in the system
  - In most cases the system is in inconsistent state at some point of the transaction, but other properties of the ACID will guarantee that it is not inconsistent for any other transaction

Transactions ACID

- Isolation
  - Simultaneous transactions in the system must be isolated from each other. They must not access inconsistent data in any state of the transaction.
  - Transaction history (schedule of the state transitions) must be serializable
    - (Pessimistic) 2 phase locking, time stamps or optimistic concurrency control

Transactions ACID

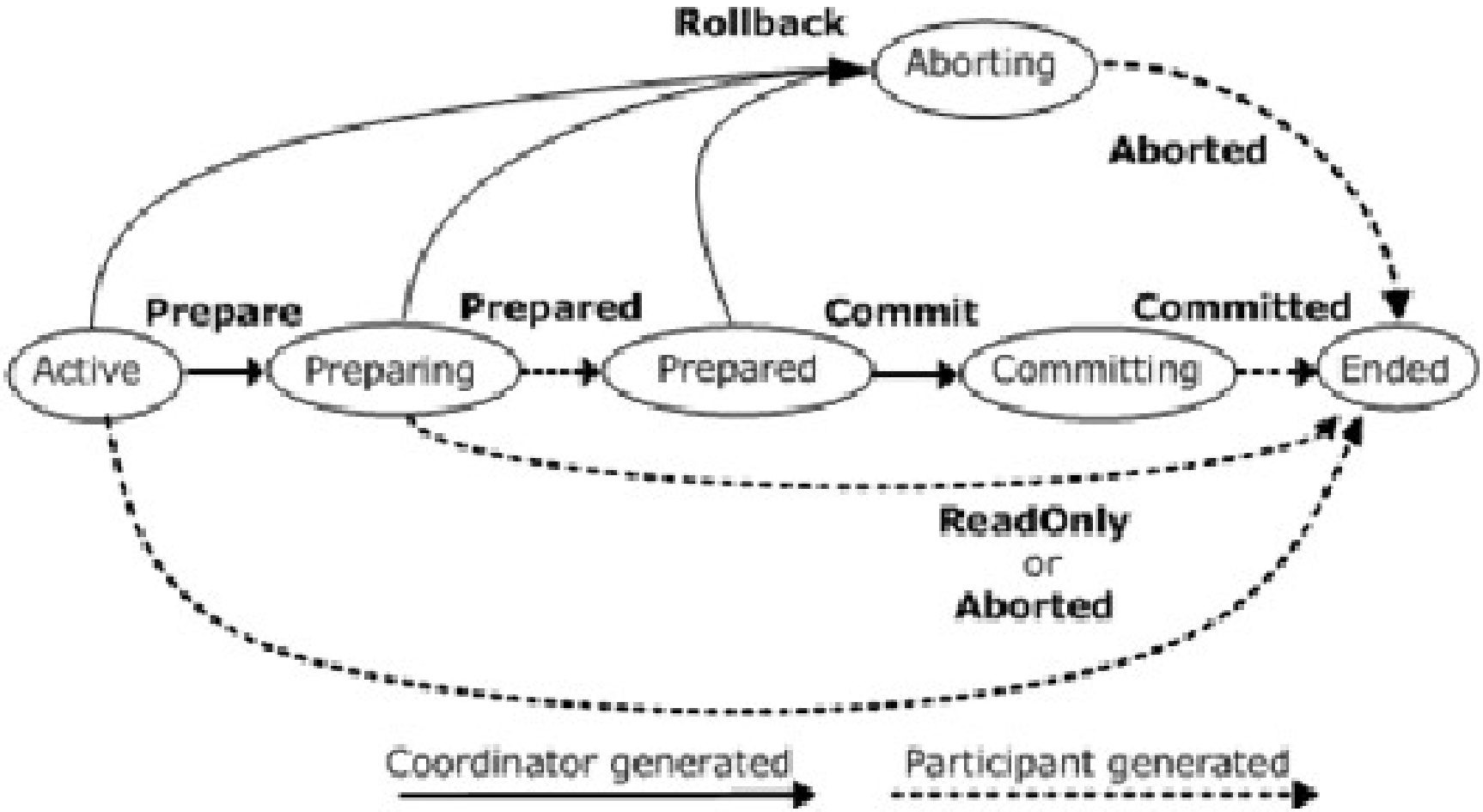
- Durability
  - Guarantee that once the transaction has been committed, it will persist, and not be undone.
  - Will survive system failure, and that the system has checked the integrity constraints and won't need to abort the transaction.

Distributed and long-living transactions

- In distributed computing transactions often cannot be handled within a RPC –type of call
  - Think about our skate-boots example
    - Transaction is committed after the actual goods arrive to the shop
    - We ”cannot” lock the service sessions (and the underlying database) for that long time
- Need for transaction coordination
  - Single or multiple coordinator processes that control the transaction and confirm or abort the transaction

2PC (Two Phase Commit)

Protocols for long living transactions 2PC



Protocols for long living transactions 2PC

- The coordinator sends vote requests to all participants
- When a participant receives the vote request, it replies by voting either Yes or No, according to whether it is able to carry out the task or not
- Participants that voted Yes start waiting for either Commit or Abort message from the coordinator. Participants that voted No can unilaterally abort

Protocols for long living transactions 2PC

- The coordinator collects all vote messages.
- If all the participants voted Yes, the coordinator decides to commit and sends Commit messages to all participants.
- Otherwise, the coordinator decides to abort and sends Abort messages to all participants that voted Yes.
- According to the received message, a participant decides to commit or abort.

Protocols for long living transactions 2PC

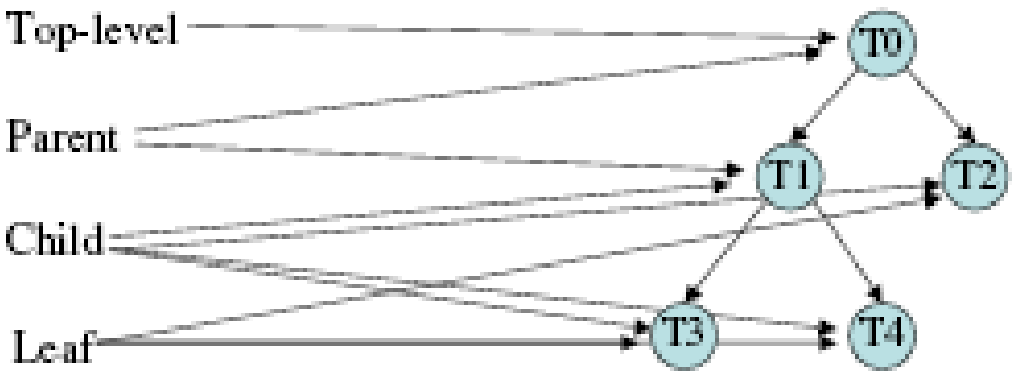
- It is possible that messages may not arrive due to failures and processes may be waiting forever! A time-out mechanism must be able to interrupt the waiting period.
- In addition, the coordinator may attach the list of the participants to the vote request message and thereby let the participants to know each other! Cooperative termination protocol.

SeCo
semantic computing

Nested Transactions

Nested transactions

- A mechanism to facilitate transactions in distributed systems (Moss 1981)
- A tree-like model with parents, children, top-level (root), leaves.



Nested transactions

- Rules
  - A parent can spawn any number of children.
  - Any number of children may be active concurrently.
  - Parent can't access data when its children are alive.
  - A child can inherit a lock held by any ancestor.
  - On child commit, its locks inherited (anti-inheritance) by parent.
  - Commit dependency: parent can commit only after all its children terminate (abort/commit).
  - Abort dependency: On parent abort, even updates of committed children are undone.
  - Updates persist only if all ancestors commit.

Nested Transactions

- Intra-transactional parallelism
  - Safe concurrency
  - Reduced response time
- Intra-transactional recovery control
  - Finer control of error handling
  - Improves availability
- System modularity
  - Composition of separately developed modules

Save Points

- Can be seen as a check point in a transaction that forces the system to save the state of the running application and return a save point identifier for future reference
- Instead of removing an entire transaction after a failure of single operation (sub-process) backward recovery can return the last valid state of the transaction saved in save point reference

Compensating Transactions

- In Saga (Garcia-Molina & Salem, 1987) model a long-living transaction is broken up into a collection of sub-transactions that interleave with other transactions.
- The results of sub-transaction can be made immediately visible after it's committed.
- Each Saga sub-transaction (Ti) is provided with a compensating transaction (Ci).
  - If the compensation is not needed saga (T1, T2, …, Tn) will execute as a sequence of transactions.
  - In case of compensation the sequence would be (T1, T2, …, Tj, Cj, …, C2, C1) where 0 ! j < n, and Ci is a predefined compensating transaction of Ti.
- The compensating transaction does not necessarily restore the state that prevailed before the execution of Ti but rather undoes operation of Ti from a semantic point of view.

Forward Recovery

- Combining the save points and compensation and recovery of failed transaction.
  - The transaction that caused the failure is aborted using the conventional roll-back.
  - Then committed transactions are undone in reversed order by their compensating counterparts until the save point is found (backward recovery).
  - Finally, the transaction is restarted from the location of found save point (forward recovery).
- The same objective can be achieved by different ways. Alternative methods of forward recovery should be considered

Transactions Summary

- ACID properties must be met
  - Not necessarily possible with single-call type of methods
- Long living transactions
  - Coordination
  - Several methods / combinations usable depending of the case
    - 2PC
    - Nested Transactions
    - Compensation
    - Forward Recovery

So we are using Web Service framework...

- WS-Transaction specification has two main paradigms for transaction control
  - Atomic Transactions (AT)
    - Pure ACID for short transactions with 2PC or similar
  - Business Activity (BA)
    - Long living transactions
    - Uses Compensation

Even if you do everything right...

- Transactions in web services world are different from their traditional environment = database transactions
- Web service is a service, not a state tracking system in IS world
  - Transactions may include or reflect to parts that are not automated
    - Sending a bill to a customer may be done by letter
    - How do you roll-back sending a letter?
  - May require a BPR (Business Process Re-engineering)