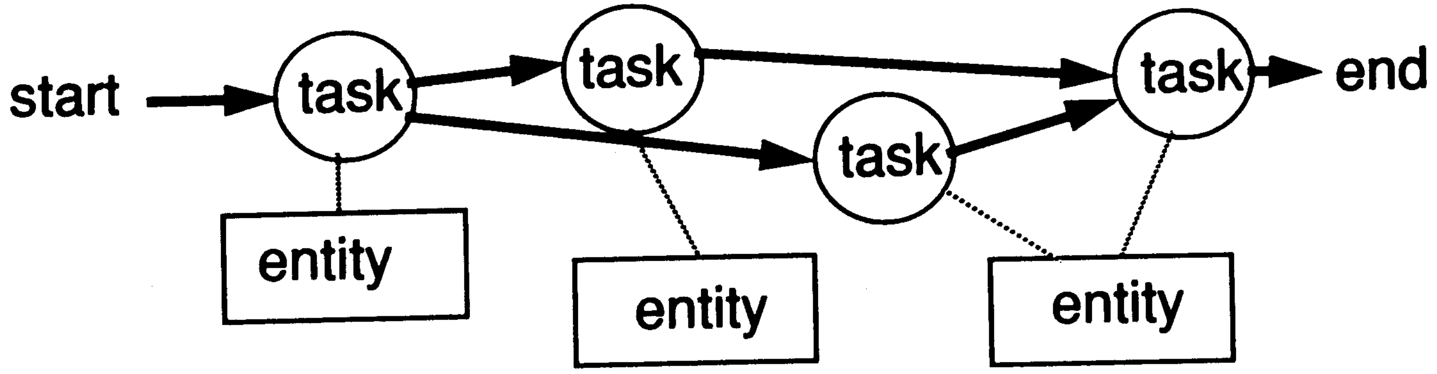
Talk Outline

- Basic concepts and specification of workflows
- Formal model for specification and scheduling of intertask dependencies
- Prototypes and applications
- Work in progress



What is a (transactional) workflow?

An activity that involves coordinated execution of multiple related tasks by different entities.

Specification issues:

- Individual Tasks:
  - task format: message, contract, form transaction
  - task structure: externally visible states of the task, initial state, termination states, significant events and *their attributes*
  - task (operation) semantics, e.g., compatibility, relaxed isolation

- Individual Entities:
  - type of entity: human, application system, DBMS
  - system properties/semantics, e.g., isolation granularity, order preservation, idempotency, monotonicity

[Sheth and Rusinkiewicz 93]

What is a (transactional) workflow?

- Task Coordination requirements:
  - intertask dependencies and data exchange
- Intra- and inter-workflow Execution requirements:
  - failure atomicity (A)
  - execution atomicity (I)
  - workflow recovery
  - inter-workflow concurrency

Workflow Examples

Environment	Application
office computing	mail routing loan processing meeting scheduling course organizing
data processing	processing a purchase order
manufacturing	product life-cycle
telecommunication	establishing or changing a service/circuit

Closely related terms/issues:
Multi-system applications [Bellcore/UofH], task flow [Dayal], long-running activities [DEC], application multi-activities [Kalinechenko], extended transaction models [Elmagarmid book], third generation TP monitor [SIGMOD93]

Related research areas [different types of tasks, different types of entities]: cooperative activity [Bellcore,..], collaborative distributed problem solving [UFL,...], DAI [DAKE, MCC,..], learning, self-adapting software agents [CMU,..]

Transactional Workflow Management

Three Components:

- Specification:

  declarative, flexible specification of tasks, dependencies, execution requirements

  extended transaction models [Elmagarmid 92] (e.g., Flexible Transaction [Elmagarmid/Rusinkiewicz et al. 90]; ACTA [Ramamritham/ Chrysanthis 91/92], dependency specifications [Klein 91]

- Scheduling:

  efficient, maximal parallelism, exploit task and system semantics

  e.g., (L.0) [Cameron et al. 91]; VPL [Kuehn et al 92]

- Executing:

  manage execution of tasks/transactions on heterogeneous, autonomous component systems

  e.g., DOL System, Narada [Rusinkiewicz/UofH], Interbase/RSI [Elmagarmid/Purdue], ESS [MCC/Carnot]

Transaction Models and Specifications

- ACID transactions and their nested derivatives

  Problems: inflexible, difficult to implement in multi-systems.

- Queued message systems and “chaining of transactions”.

  Problems: insufficient control over transaction properties, interactions among concurrent activities difficult.

- Extended/Relaxed Transaction Models:
  - Sagas and Nested Sagas [Garcia-Molina et al. 88, 90]
  - ConTracts [Reuter 89]
  - Flexible Transactions [Elmagarmid et al 90, Rusinkiewicz et al 90]
  - Multi-transaction Activities [Garcia-Molina et al. 90]
  - Long-Running Activities [Dayal et al. 91]
  - Relaxed transactions in Carnot [Cannata 91]
  - The DOM project [Buchmann et al 92]
  - Open Nested Transactions [Weikum & Schek 92] and Others (e.g., in [Elmagarmid 92], SIGMOD93)

[Ref: Rusinkiewicz, Dayal]

Significant Events

Significant event types for <u>database applications</u>: *st, ab, pr, cm*

Possible attributes of an event type:

- Forcible: the system can always force the execution
- Rejectable: the system can always reject the event
- Delayable: the system can delay execution of the event (every non-real-time significant events are delayable)

Event	Forcible?	Rejectable?	Delayable?
cm	N	Y	Y
ab	Y	N	N
pr	N	N	N
st	Y	Y	Y

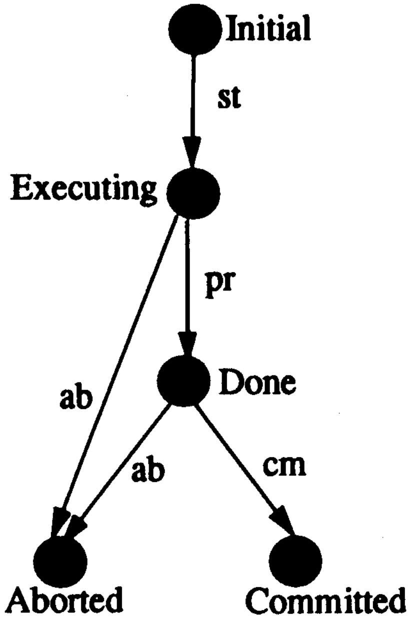
Usual attribute assignments for transactions in database applications and DBMSs

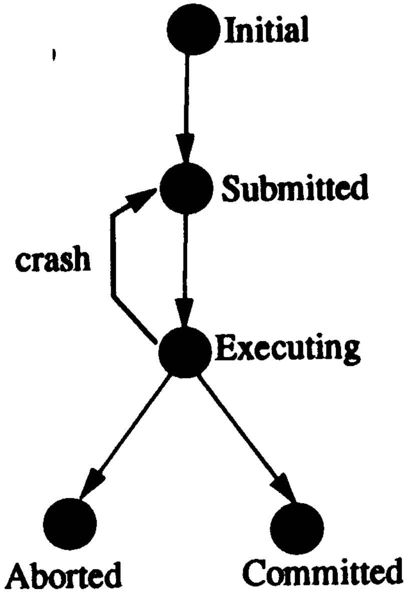
[Attie et al 93]

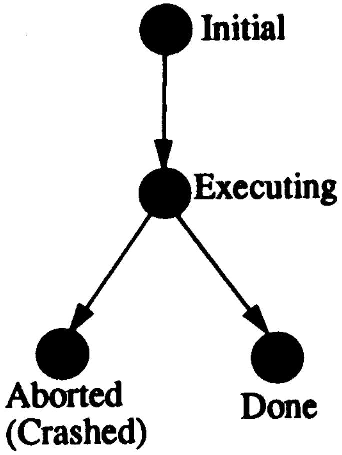
Task Skeleton

- different skeleton depending on application and the system
  - different states (e.g., no precommit)
  - different significant events submitted by application/user and system

Examples:







[Rusinkiewicz & Sheth 93]

Intertask Dependencies

Preconditions for initiating each scheduler-controllable transition in a task.

Klein's primitives [KL91]:

- Order Dependency: $e_1 < e_2$.
  If both e_1 and e_2 occur, then e1 precedes e2.
  Alternatively, in CTL: if e_2 occurs, e_1 cannot occur subsequently.
  Formally specified as: $\mathrm{AG}[e_2 \Rightarrow \mathrm{AG}{\sim}e_1]$
- Existence Dependency: $e_1 \rightarrow e_2$.
  If event e_1 occurs sometimes, then event e_2 also occurs sometimes.
  Alternatively, there is no computation such that e_2 does not occur until a state s is reached where s satisfies [e_1 is executed in s, and subsequently, e_2 never occurs].
  Formally specified as: ${\sim}\mathrm{E}[{\sim}e_2 \,\mathrm{U}\, (e_1 \wedge \mathrm{EG}{\sim}e_2)]$

– Conditional Existence Dependency [KL91]: $e_1 \rightarrow (e_2 \rightarrow e_3)$]

Examples from multidatabase transaction models:

- Commit Dependency [CR92]: $cm_B < cm_A$
- Abort Dependency [CR92]: $ab_B \rightarrow ab_A$

[Attie et al 93, Rusinkiewicz & Sheth 93]

Enforceable Dependencies

- Dependencies may not be enforceable.

  For example, *ab(A) -> cm(B)*

- Event attributes determine whether a dependency is enforceable. For example,
  - e_1 -> e_2 is run-time enforceable if

    rejectable(e_1) [delay *e1* until *e2* is submitted, reject *e1* if task 2 terminated without submitting *e2*],

    or forcible(e_2) [force execution of e_2 when e_1 is accepted for execution].
  - $e_1 < e_2$ is run-time enforceable if

    rejectable(e_1) [let e_2 be executed when it is submitted, thereafter reject e_1 if submitted],

    or delayable(e_2) [delay e_2 until either e_1 has been accepted for execution, or task 1 has terminated without issuing e_1].

Beyond Dependencies --
Task Coodination Requirements

Statically -- a precondition for starting a task or initiating a transition in a task.

Preconditions may be specified with dependencies involving:

- execution states of other tasks
- optput values of other tasks
- external variables (events outside the workflow, time,..)

E.g., execution dependencies, data/value dependencies, temporal dependencies in Flexible Transactions [Elmagarmid et al 90], ConTracts [Reuter 89], Multitransactions [Garcia-Molina et al 90], Multidatabase Transactions [Rusinkiewicz et al 92]....

Dynamically--

Created when executing a workflow.

Long-running activities [Dayal et al 91], Polytransactions [Rusinkiewicz and Sheth 91].

[Rusinkiewicz and Sheth 93]

Scheduler Operation (An Example)

Consider only $e_1 < e_2$ and $e_1 \text{->} e_2$ dependencies, where both e_1 and e_2 are rejectable (e.g., all dependencies for SAGAs can be expressed using these). Corresponding automata $A_<$ and $A_{->}$.

- e_1 is submitted.
  - $a(e_1)$ in $A_<$. No path in $A_{->}$ with e_1. e_1 added to pending set.
- e_2 is submitted.
  - A->: $a(e_2);a(e_1)$ and $a(e_2)|||a(e_1)$.
  - *a-closure* forces searching A< for a path that accepts both e_1 and e_2. Only such path is $a(e_1);a(e_2)$ which is not order-consistent with $a(e_2);a(e_1)$.
  - Viable pathset is $\{a(e_1);a(e_2), a(e_2)|||a(e_1)\}$.
  - Partial order consistent with this is e_1 and then e_2.

ion/Task and Systm Semantics

/Task, System)	Impact (CCon. Control, R: Recovery)
appl)	fewer exclive locks (CC)
	no global ammitment (CC)
gorousness (sys)	early relea of locks (CC)
	resubmit trsactions (R)
	roll-forwarecovery (R)

[Jin et al 93]

Conclusions

- Gained detailed understanding on issues of transaction workflows.
- Studied and demonstrated applicability of relaxed multidatabase transaction.
- Developing a generic model workflow.
- **Developed formal approach to specifying and executing workflows.**
- Completed large-scale prototype.
  Demonstrated with a real application.
  Deployment being considered.

External Publications

E. Jane Cameron, Linda Ness, and Amit Sheth, "A Universal Executor for Flexible Transactions which Achieves Maximal Parallelism," in the *Proc. of the 1st Intl. Workshop on Interoperability in Multidatabase Systems* (IMS '91), April 1991.

Authors1, "Executing Multidatabase Transactions," in the Proc. of the 25th Intl. Conf. on Systems Sciences, January 92.

Authors1, "Using Flexible Transactions to Support Multi-system Applications", presented at the *1992 Workshop on Heterogeneous Databases and Semantic Interoperability,* U S West, Boulder, February 1992.

Authors1, "Applying Multidatabase Transactions for Service Order Processing", in the *Proc. of the 18th Intl. Conf. on Very Large Databases* (VLDB'92), August 1992.

Authors2, Concurrency Control and Recovery of Multidatabase Work Flows in Telecommunication Applications. in the *Proc. of ACM SIGMOD Conference*, May 1993.

D. Woelk, P. Attie, P. Cannata, G. Meredith, A. Singh, M. Singh, and C. Tomlinson, "Task Scheduling using Intertask Dependencies in Carnot," in the *Proc. of ACM SIGMOD Conference*, May 1993.

P. Attie, M. Singh, M. Rusinkiewicz, and A. Sheth, "Specifying and Enforcing Intertask Dependencies," MCC Technical Report # Carnot-245-92, November 1992. Abridged version in the *Proc. of the 19th Intl Conf. on Very Large Data Bases*, August 1993.

A. Sheth and M. Rusinkiewicz, "On Transactional Workflows," in Data Engineering Bulletin, 16 (2), June 1993.

Authors1 (different orders): M. Ansari, L. Ness, M. Rusinkiewicz, A. Sheth
Authors2: W. Jin, L. Ness, M. Rusinkiewicz, A. Sheth