SERENE 2008

Addressing Degraded Service Outcomes and Exceptional Modes of Operation in Behavioural Models

Sadaf Mustafiz, Jörg Kienzle

School of Computer Science
McGill University, Montreal, Canada

{sadaf, joerg} @cs.mcgill.ca

Andrey Berlizev

Laboratory of Advanced Software Systems
University of Luxembourg, Luxembourg

andrey.berlizev@uni.lu

Motivation

- Dependability not usually addressed in current mainstream software development methods
- Lack of methods for elicitation and specification of degraded service outcomes and modes of operation
- Not addressed in current modelling formalisms
- Necessary to identify, specify, and analyze dependability concerns during the early stages of software development
- Need to precisely define system behaviour in exceptional situations
- Need to satisfy users and maintain system safety and reliability

Software and cathedrals are much the same - first we build them then we pray! [Sam Redwine, Jr.].

McGill

Overview

- Background
- Degraded service outcomes
- Modes of operation
- Requirements Engineering and DREP
- Modelling degraded outcomes
- Modelling modes
- Related work
- Conclusion

} Elevator Control System

McGill

Dependability

- **Dependability** is that property of a computer system such that trust can justifiably be placed on the service it delivers[1].
  - Reliability - *aptitude to provide continuity of service*
  - Safety - *lack of catastrophic failures*

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. Dependable and Secure Computing, IEEE Transactions on, 1(1):11-33, Jan.-March 2004. .

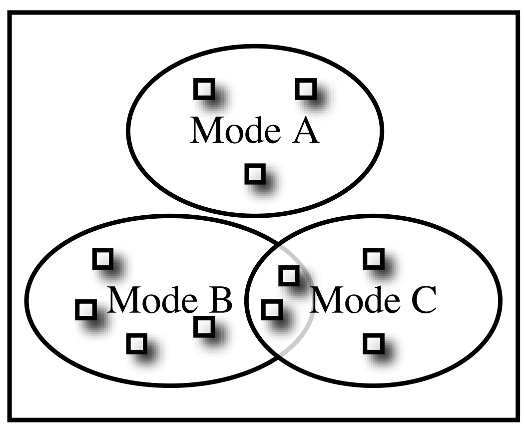
McGill

Degraded Service Outcomes

- System providing a set of services
- Exceptional situation does not allow the system to complete the task at hand
- Discuss with stakeholders
- Handle current situation
- Provide partial or degraded service
- Provision of degraded outcome better than complete service failure
  - success, failure, degraded success

Exceptional Modes of Operation

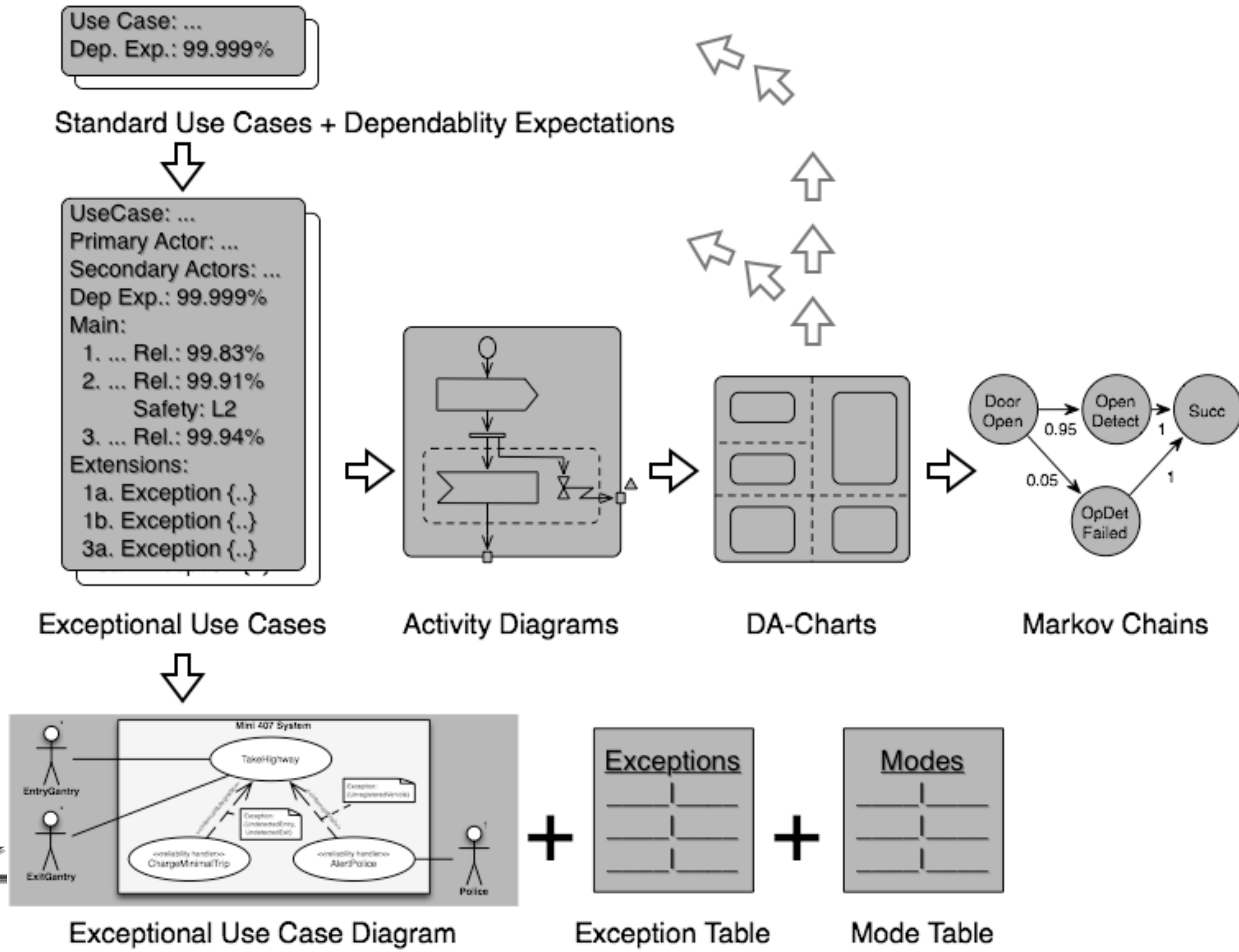
- Provision of some services not possible due to exceptional situation
- Do not offer services that cannot be provided
- Mode determines the set of services currently offered
- Switch into a different mode



Requirements Engineering

- Requirements Development
  - discovery and elicitation
  - definition and specification
  - analysis of the requirements
  - system validation against the requirements
- Dependability-Focused Requirements Engineering Process (DREP)

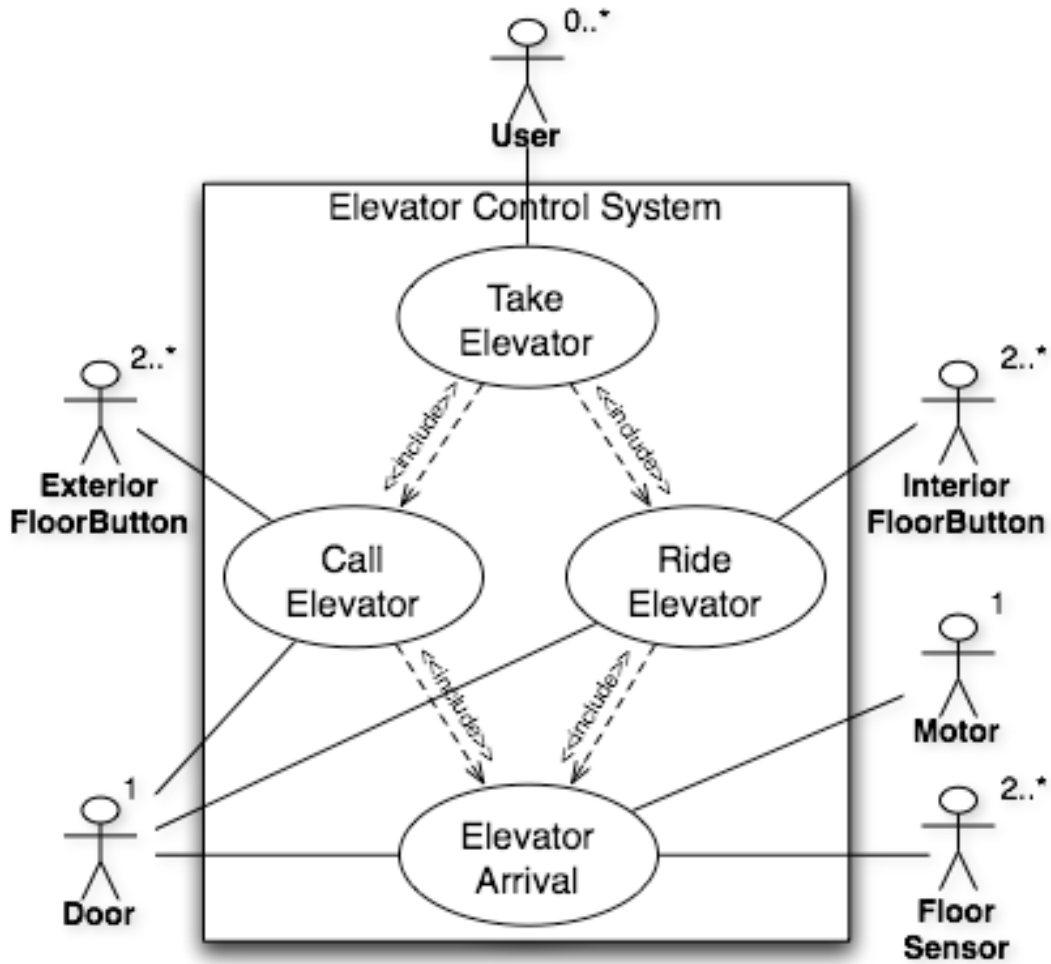
Model-based DREP



Dependability in Use Cases

- Use Cases capture interactions between the system and the environment to achieve user goals
- Error detection - detection of exceptional situations by means of secondary actors (*e.g.*, sensors) and timeouts
  - situation that threatens the successful completion of the user goal (reliability)
  - situation puts user in danger (safety)
- System recovery - describing interactions with the environment required to
  - continue to deliver the current service (reliability)
  - offer a degraded service (reliability)
  - take actions that prevent a catastrophe (safety)

Elevator Control System Case Study



Elevator Arrival Exceptional Use Case (1)

Use Case: ElevatorArrival

Intention: System wants to move the elevator to the *User*'s destination floor.

Main Success Scenario:

1. System asks Motor to start moving towards the destination floor.
2. Movement Sensor informs System that cabin is moving.
3. Approaching Floor Sensor informs System that the cabin is approaching destination floor.
4. System requests Motor to stop.
5. Floor Sensor informs System that elevator is stopped.
6. System requests Door to open.
7. Door Sensor informs System that door is open.

Use case ends in *<< success >>* *ReachedDestination*.

Elevator Arrival Exceptional Use Case (2)

Extensions:

2a. Exception {MotorStartFailure}

  2a.1 Use case continues at step 1.

    2a.1.a Retried 3 times.

      Use case ends in << *failure* >> *MotorFailure*.

3a. Exception{MissedFloor}

  3a.1 System chooses to approach a neighboring floor. Use case continues in << *degraded* >> *DifferentFloor* at step 1.

5a. Exception{MotorFailure}

  5a.1 Use case ends in << failure >> MotorFailure.

7a. Exception{DoorStuckClosed}

  7a.1 Use case continues at step 6.

    7a.1a Retried 3 times.

      Use case ends in << *failure* >> DoorStuckClosed.

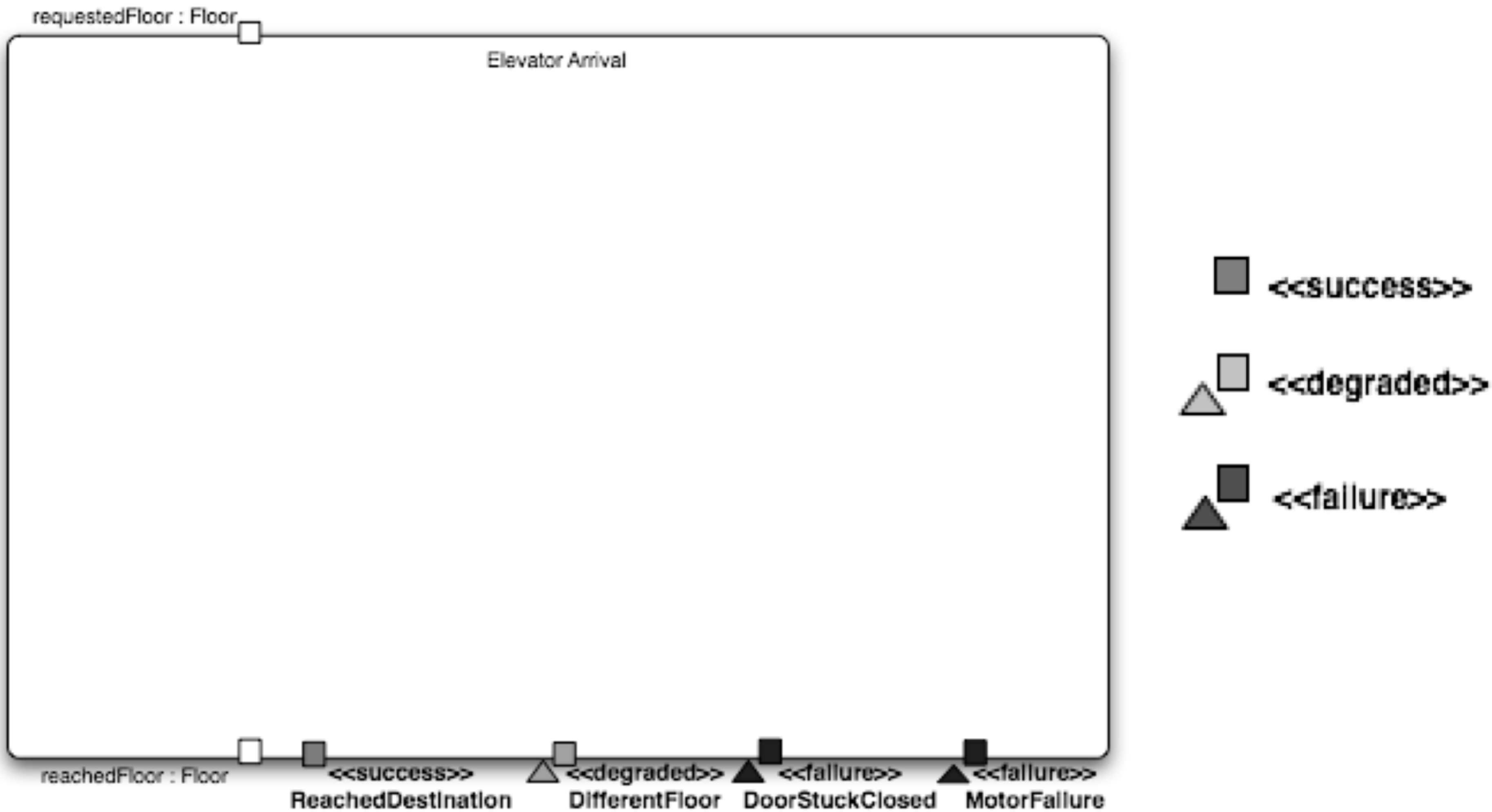
Outcomes in the EA Use Case

- Success << *success* >>
  - Reached Destination
- Degraded success << *degraded* >>
  - Different Floor
- Failure << *failure* >>
  - Motor Failure
  - Door Stuck Closed

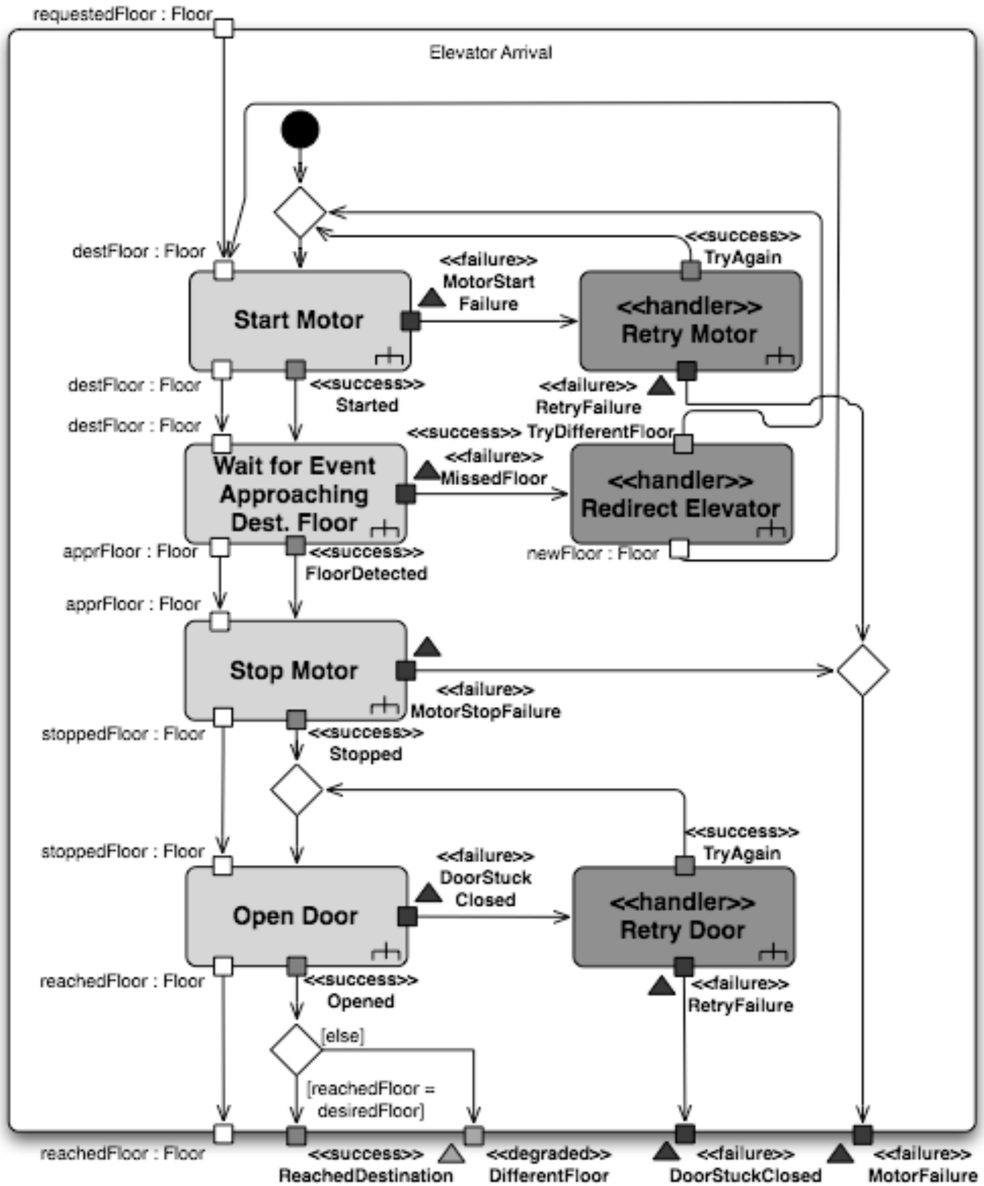
Activity Diagrams

- Models workflow behaviour
- Emphasizes the sequence and conditions
- Models activities
  - Actions
  - Sending or receiving messages
  - Control flow
  - Object flow
- Hierarchical structuring

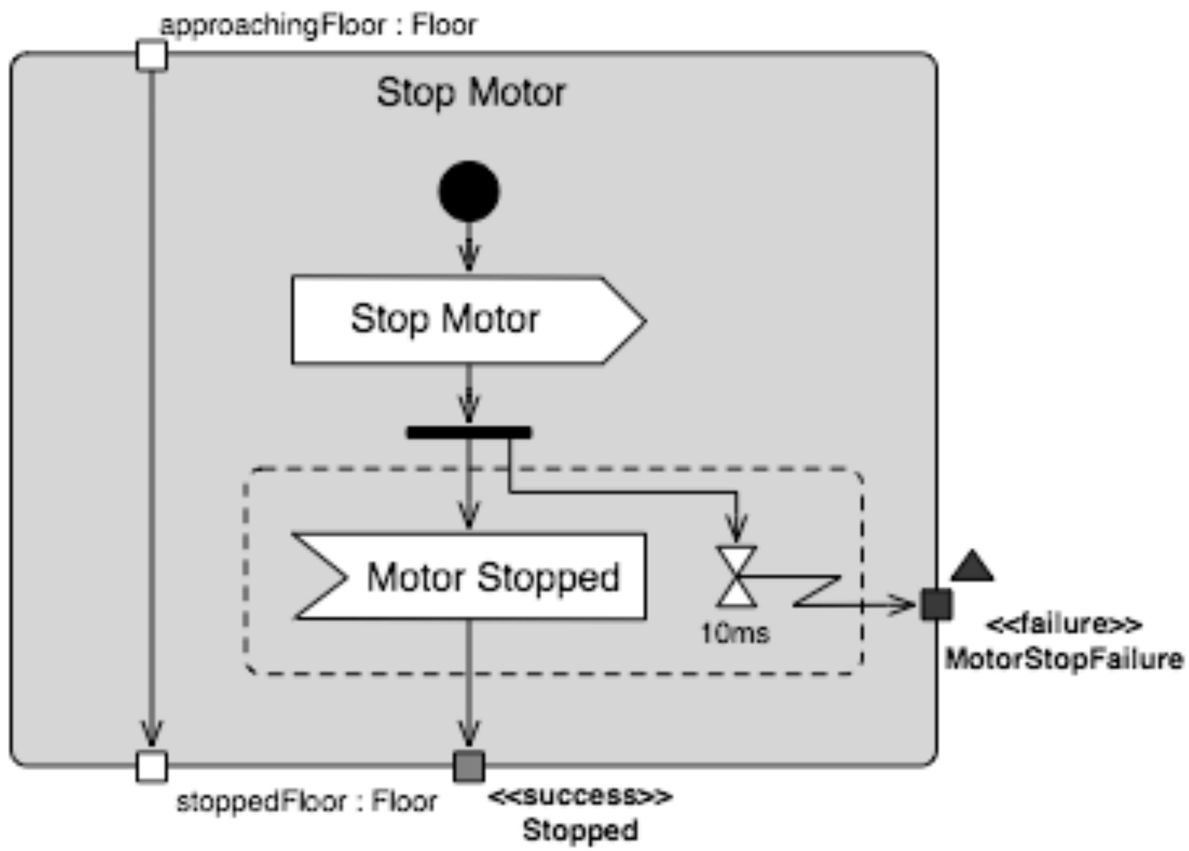
Outcomes in Activity Diagrams (1)



Outcomes in Activity Diagrams (1)



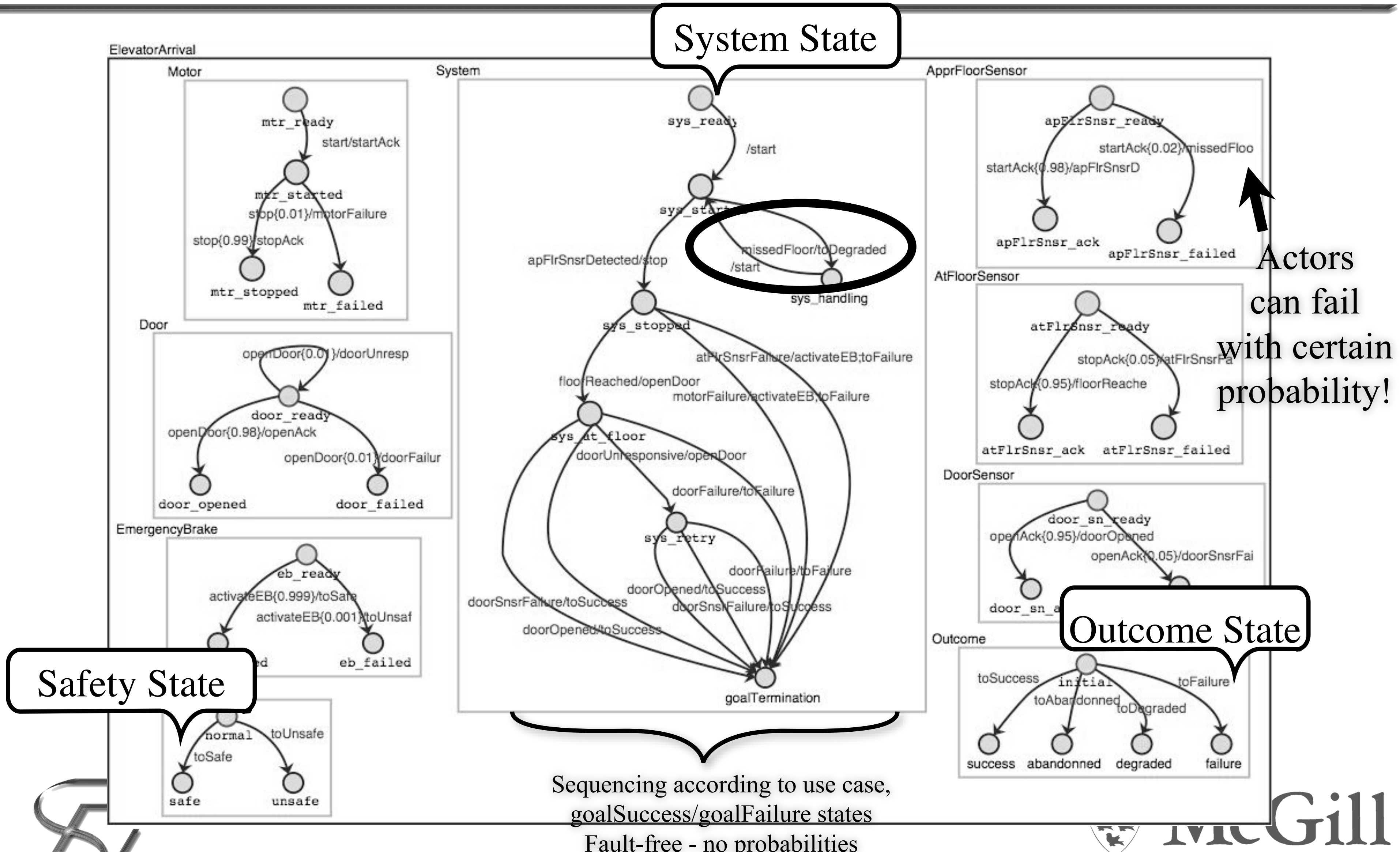
Outcomes in Activity Diagrams (2)



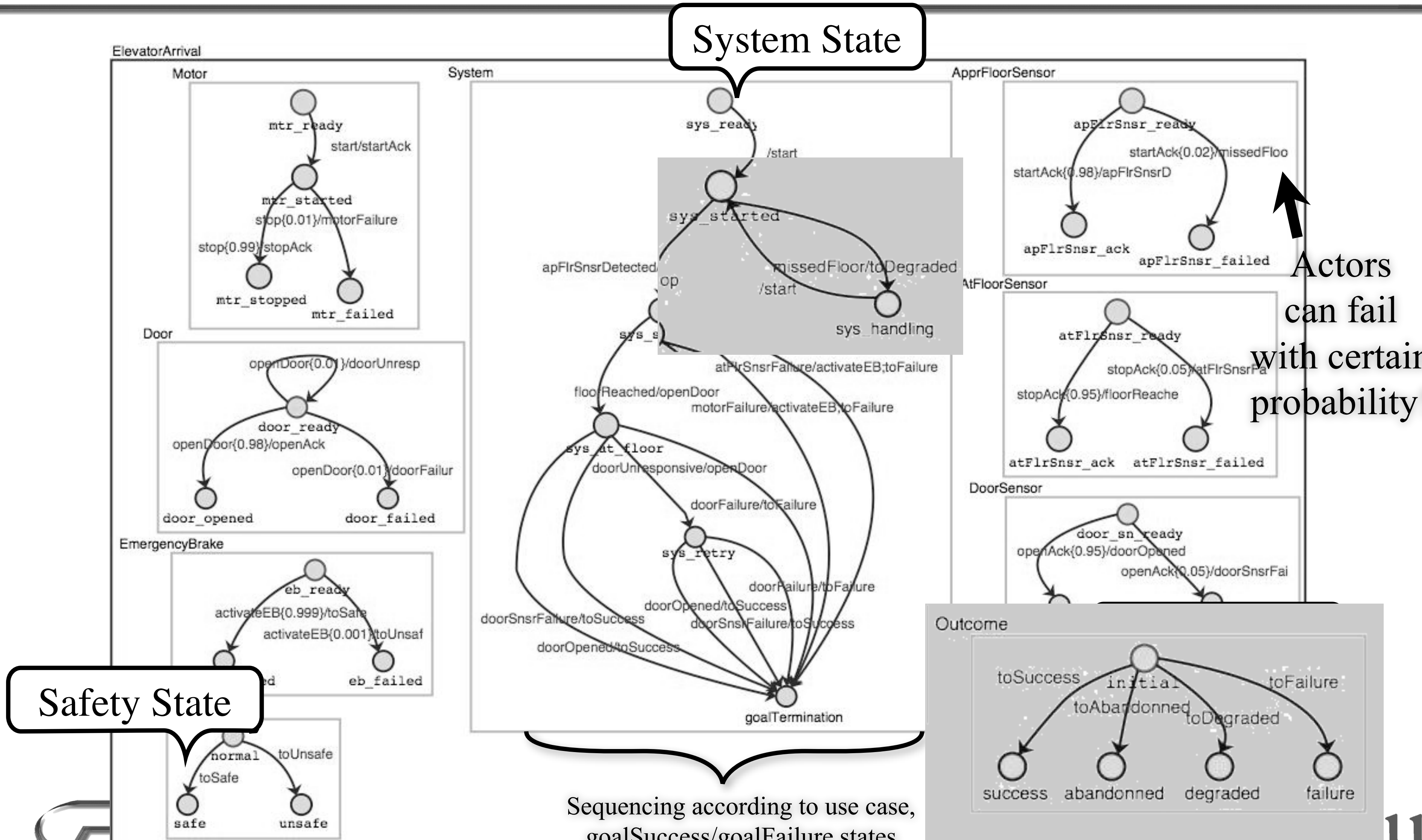
Statecharts

- Modelling of reactive system behaviour
- Used for documentation, analysis, and simulation
- DA-Charts
  - probabilistic extension of statecharts
  - model-driven assessment for use cases
  - tool support to compute the probability of ending in a safe state or of completing a goal

Outcomes in DA-Charts



Outcomes in DA-Charts

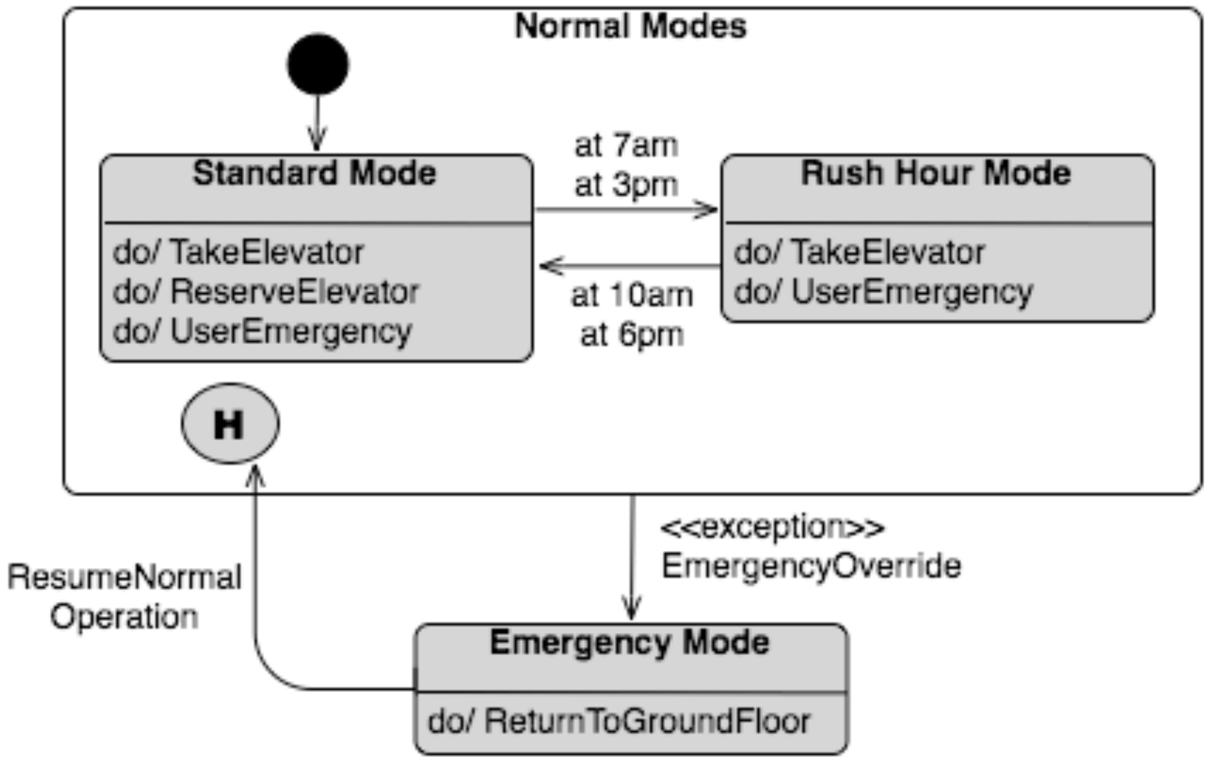


Sequencing according to use case,
goalSuccess/goalFailure states
Fault-free - no probabilities

Exceptional Modes

- Normal mode
  - services available and correctly functioning
- Degraded mode
  - offers limited services
  - degraded QoS provided
- Emergency mode
  - normal services suspended
  - emergency services offered
- Restricted mode
  - subset of normal services offered
  - emergency services available

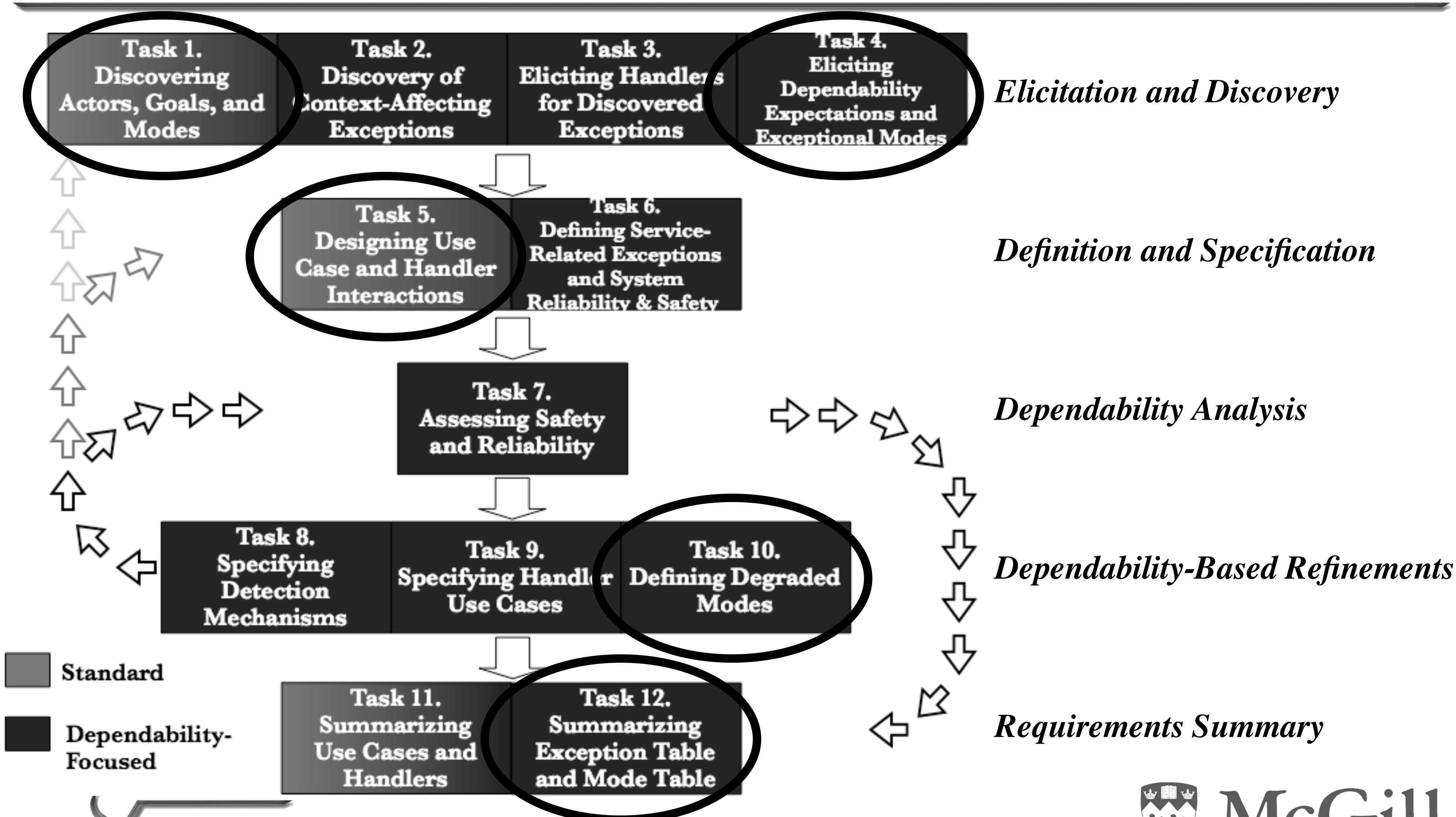
Modelling Modes



Mode Table

ID	Title	Description	Expected Reliability	Expected Safety	Other Modes

Task-based DREP



Related Work

- CORA
  - proposed by Berlizev and Guelfi
  - analysis method for embedded system modelling
  - activity model, domain model, operation model
  - degraded service outcomes addressed in activity models
- Work on degraded modes of operation proposed by Srivasta et al, Shea and Johnson, Lygeros et al.

Conclusion

- Introduced concepts of degraded service outcomes and exceptional modes of operation
- Have shown how to model the concepts in different formalisms
  - use cases, activity diagrams, statecharts
- Integration in requirements engineering
  - Illustrated by incorporating concepts in DREP
  - Elicitation and specification of outcomes
    - exceptional use cases to elicit
    - exceptional activity diagrams to specify
    - DA-Charts to analyze
  - Elicitation and specification of modes
- Leads to a complete requirements specification document

References (1)

1. I. F. Alexander. Misuse cases: Use cases with hostile intent. *IEEE Software*, 20(1):58–66, 2003.
2. A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33, Jan.-March 2004.
3. A. Berlizev and N. Guelfi. CORRECT Analysis for Embedded System Modeling: An Outcome of East-West Scientific Cooperation. In *SEESE 08*, pages 23 – 30. Proceedings of the IEEE, 2008.
4. J.-C. Geffroy and G. Motet. *Design of Dependable Computing Systems*. Kluwer Academic Publishers, 2002.
5. D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, May 1988.
6. C. Larman. Applying UML and Patterns: *An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice Hall, 2002.
7. P. A. Lee and T. Anderson. Fault tolerance principles and practice. In *Dependable Computing and Fault-Tolerant Systems*. Springer Verlag, 1990.
8. J. Lygeros, D. N. Godbole, and M. E. Broucke. Design of an extended architecture for degraded modes of operation of AHS, May 26 1995.

McGill

References (2)

9. S. Mustafiz and J. Kienzle. DREP: A requirements engineering process for dependable reactive systems. In M. Butler, C. Jones, A. Romanovsky, and E. Troubitsyna, editors, *Methods, Models, and Tools for Fault Tolerance*. 2008. (To be published)
10. S. Mustafiz, X. Sun, J. Kienzle, and H. Vangheluwe. Model-driven assessment of system dependability. *Software and Systems Modeling (SoSym)*, March 2007.
11. Object Management Group. *Unified Modeling Language: Superstructure*, October 2004.
12. C. Shea and C. Johnson. The contribution of degraded modes of operation as a cause of incidents and accidents in air traffic management. In *Proceedings of the 25th ISSC*, pages 616–626, 2007.
13. A. Shui, S. Mustafiz, J. Kienzle, and C. Dony. Exceptional use cases. In *MoDELS*, volume 3713 of LNCS, pages 568–583. Springer, 2005.
14. D. Srivastava and P. Narasimhan. Architectural support for mode-driven fault tolerance in distributed applications. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7, 2005.

Thank you!

??? Questions ???

Contact: sadaf@cs.mcgill.ca