Rational. software

IBM® Rational® Unified Process®

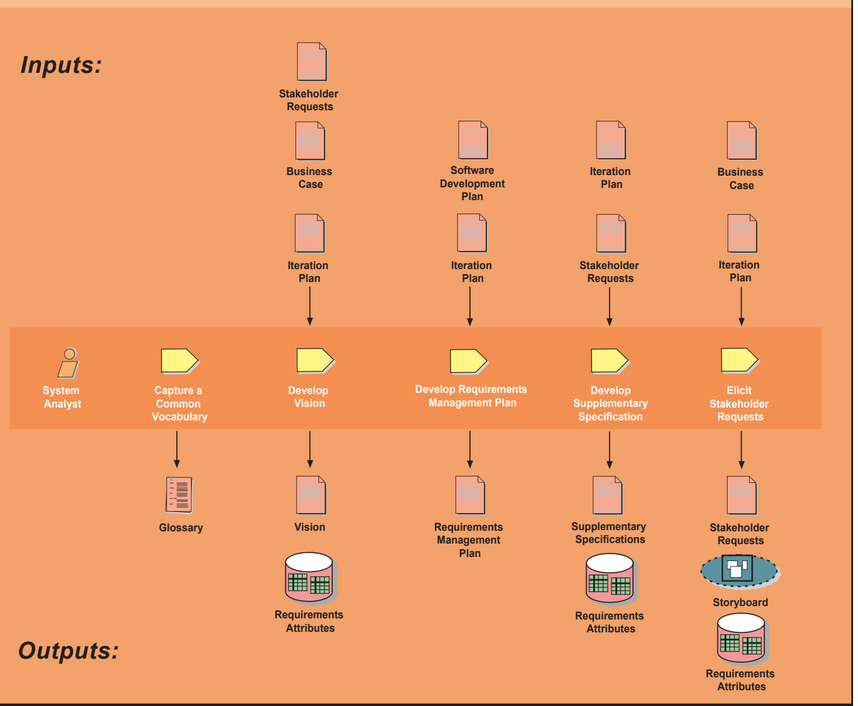
for Model-Driven Systems Development

Included in IBM® Rational® Method Composer: Helping companies implement effective processes for successful software and systems projects

GATHER SOURCE REQUIREMENTS

Understand the stakeholders, collect and prioritize requests on what needs the system should fulfill. Define the overall vision of the system, including the problem to be solved, the scope/boundary of the system, the system's key features (requirements), and any constraints.

Inputs:

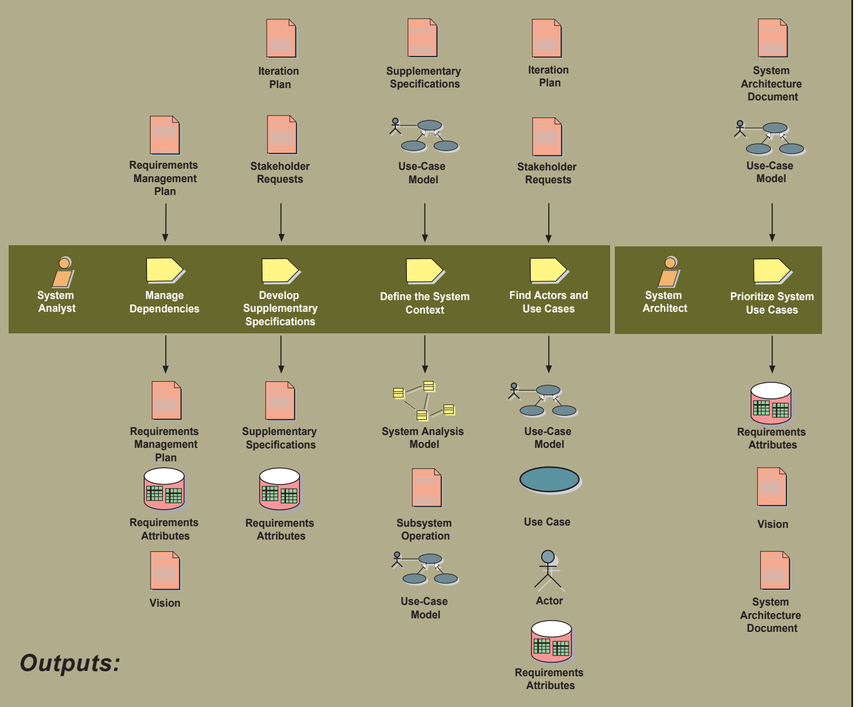


Outputs:

ESTABLISH SYSTEM CONTEXT

Define, model, and create a top-level collaboration (use-case model) showing the system, its interfaces, and its relationships with its actors, including the external I/O entities that flow between actors and the system.

Inputs:

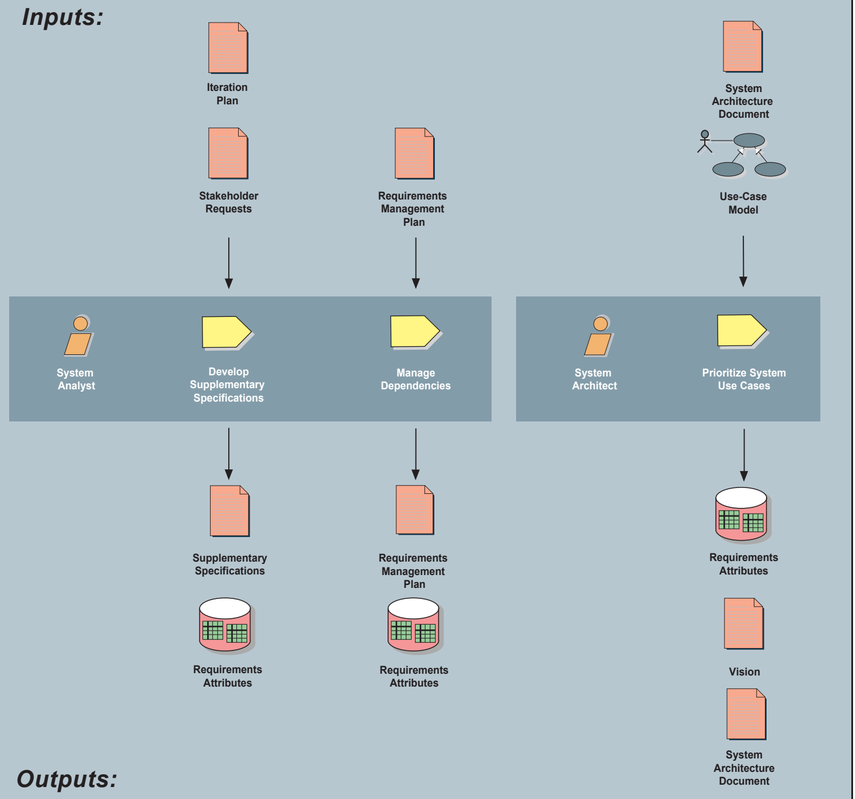


Outputs:

REFINE THE SYSTEM DEFINITION

Develop supplementary requirements (that do not apply to specific use cases), define and manage dependencies or traceability between requirements, and further detail the system use cases.

Inputs:

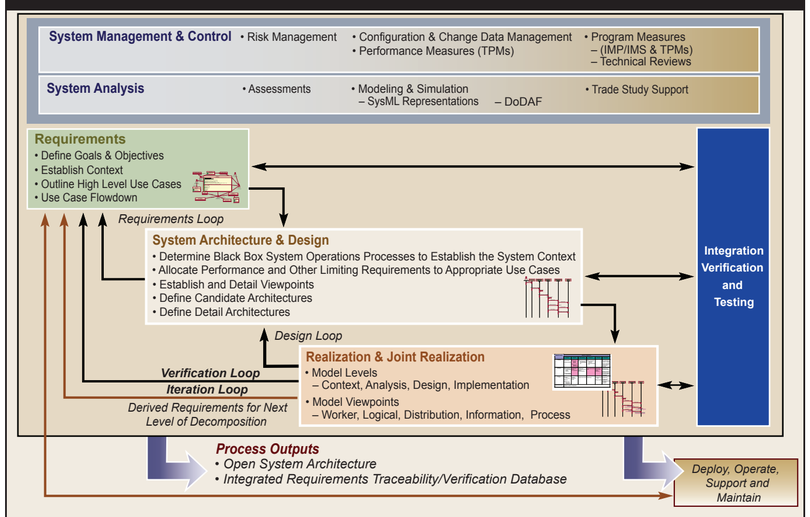


Outputs:

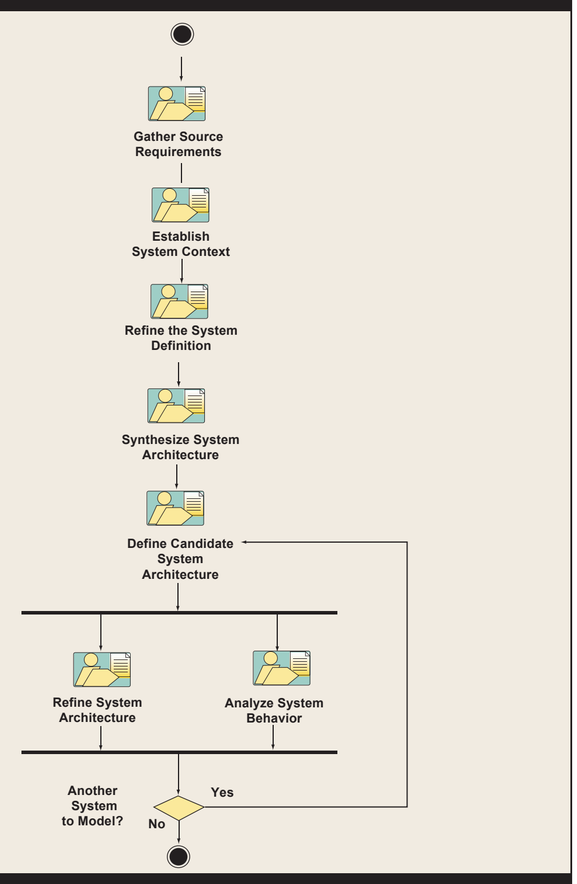
SIX PRINCIPLES OF SYSTEMS DEVELOPMENT

- Decompose systems, not requirements
- Enable both separation and integration of concerns
- Systems and components collaborate; so should development teams
- Specifications flow up and down the Architecture
- Base the life cycle on removing risk and adding value
- Development organization should reflect product Architecture

MODEL-DRIVEN SYSTEMS DEVELOPMENT (MDSD)



MDSD CAPABILITY PATTERN



MDSD CORE VALUES

Systems need to be **developed incrementally.** *This enables effective feedback loops, ongoing learning, tactical and strategic flexibility, improved oversight and decision making.*

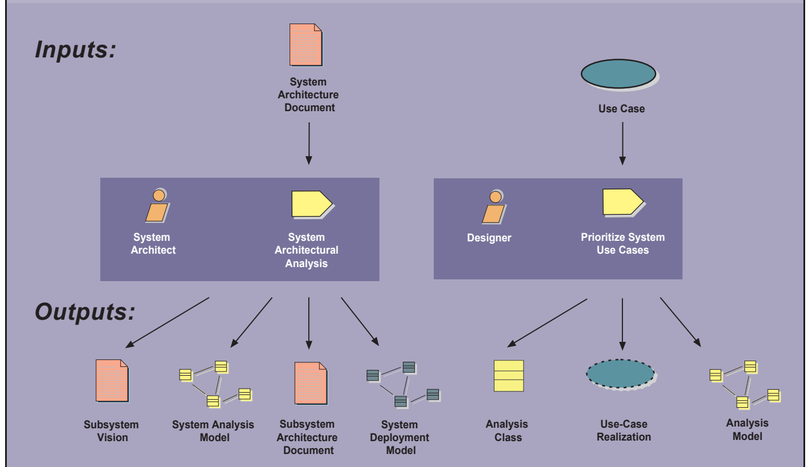
Systems should be **evolved though a set of interrelated and well-defined models** based on industry standards, such as SysML, UML, UPDM, DoDAF, etc. *This enables interoperability between tools, improved reuse, semantically correct requirements, testable requirements, architectural integrity, as well as early testing and/or simulation of partially completed systems.*

Systems-of-systems should be developed as a **collaboration between parts**, versus the traditional notion of functional decomposition. *This allows tradeoffs for each part to be negotiated in context of the overall system, as well as better integration of subordinate systems across all lifecycle activities.*

SYNTHESIZE SYSTEM ARCHITECTURE

Construct and assess a System Architecture Proof-of-Concept, with the objective of showing that there exists, or is likely to exist, a solution which will satisfy the architecturally significant requirements, showing that the system, as envisioned, is feasible.

Inputs:

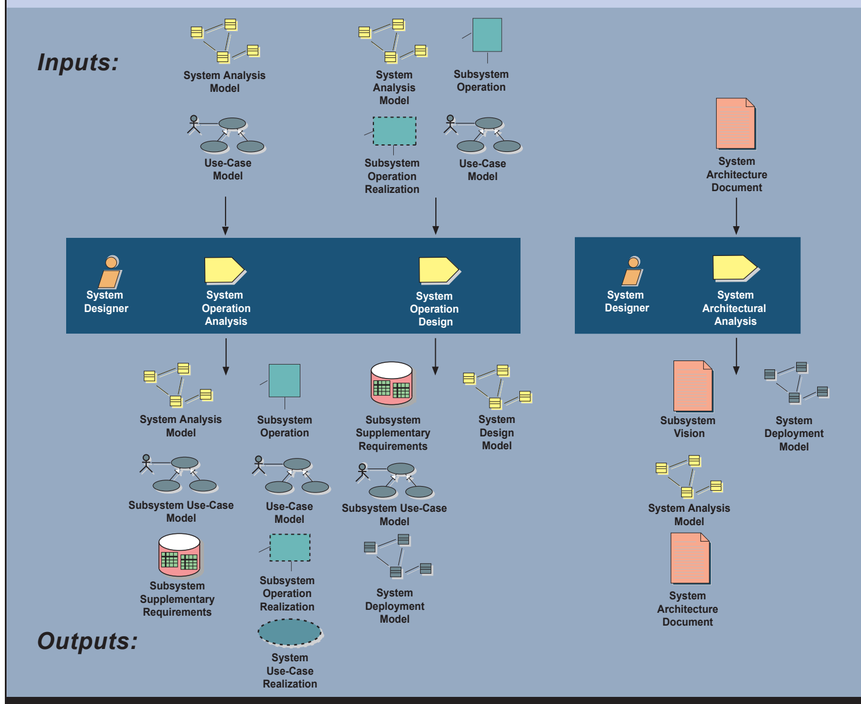


Outputs:

DEFINE CANDIDATE SYSTEM ARCHITECTURE

Create an initial sketch of the system architecture.

Inputs:

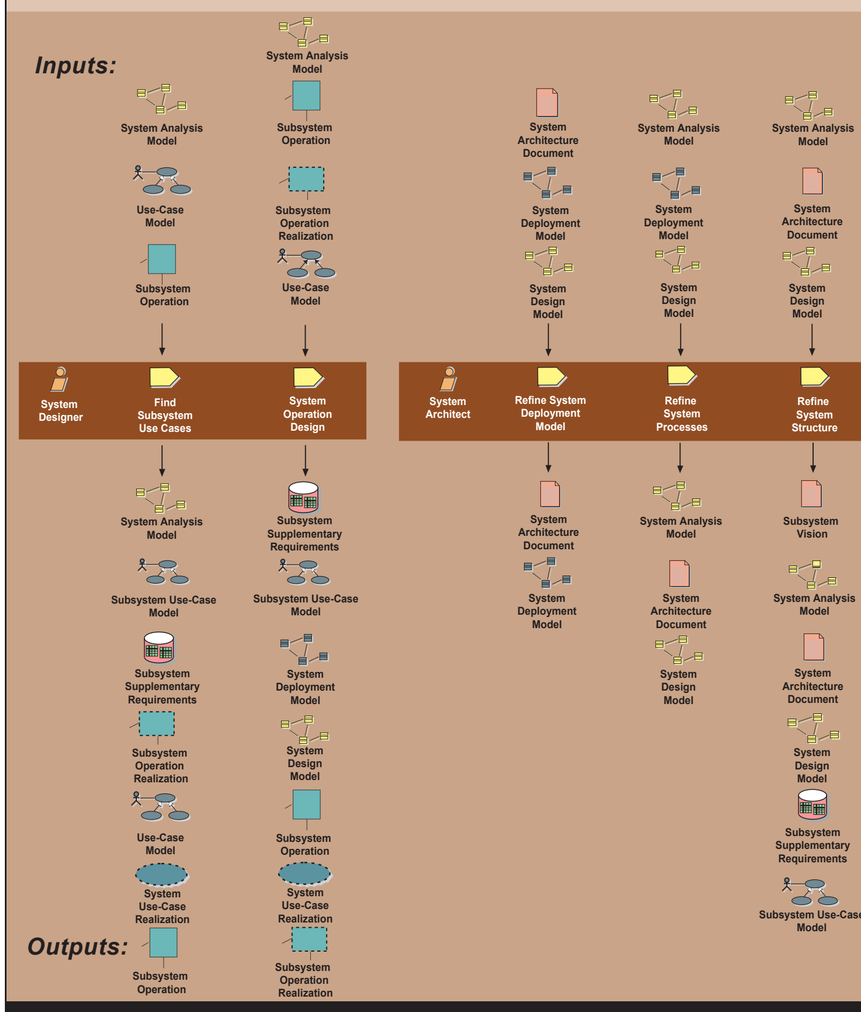


Outputs:

ANALYZE SYSTEM BEHAVIOR

Transform the behavioral descriptions provided by the requirements into a set of elements upon which the design can be based.

Inputs:

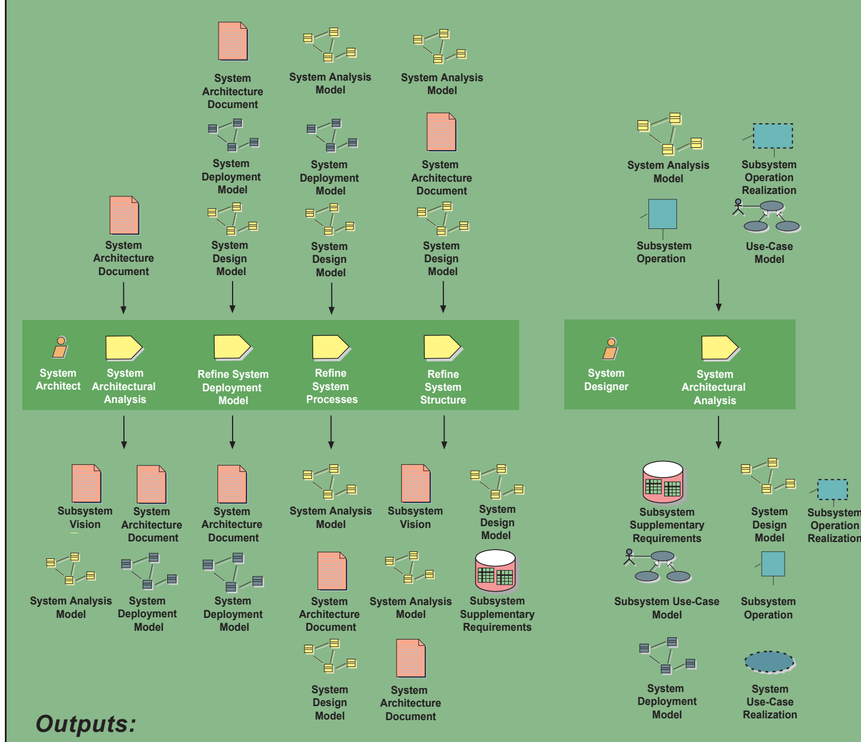


Outputs:

REFINE SYSTEM ARCHITECTURE

Maintain the system architecture aligned with the changes due to either requirements modifications or design refactoring and reusability issues.

Inputs:



Outputs:

IBM®