TATA CONSULTANCY SERVICES

TATA

Addressing Scalability Challenges in SKA Monitoring & Control

N. Swaminathan & Harrick Vin

Systems Research Lab (SRL), Tata Consultancy Services (TCS)

25 YEARS OF innovation TRDDC 1981 - 2006

TATA RESEARCH DEVELOPMENT AND DESIGN CENTRE

Experience certainty. IT Services Business Solutions Outsourcing

Agenda

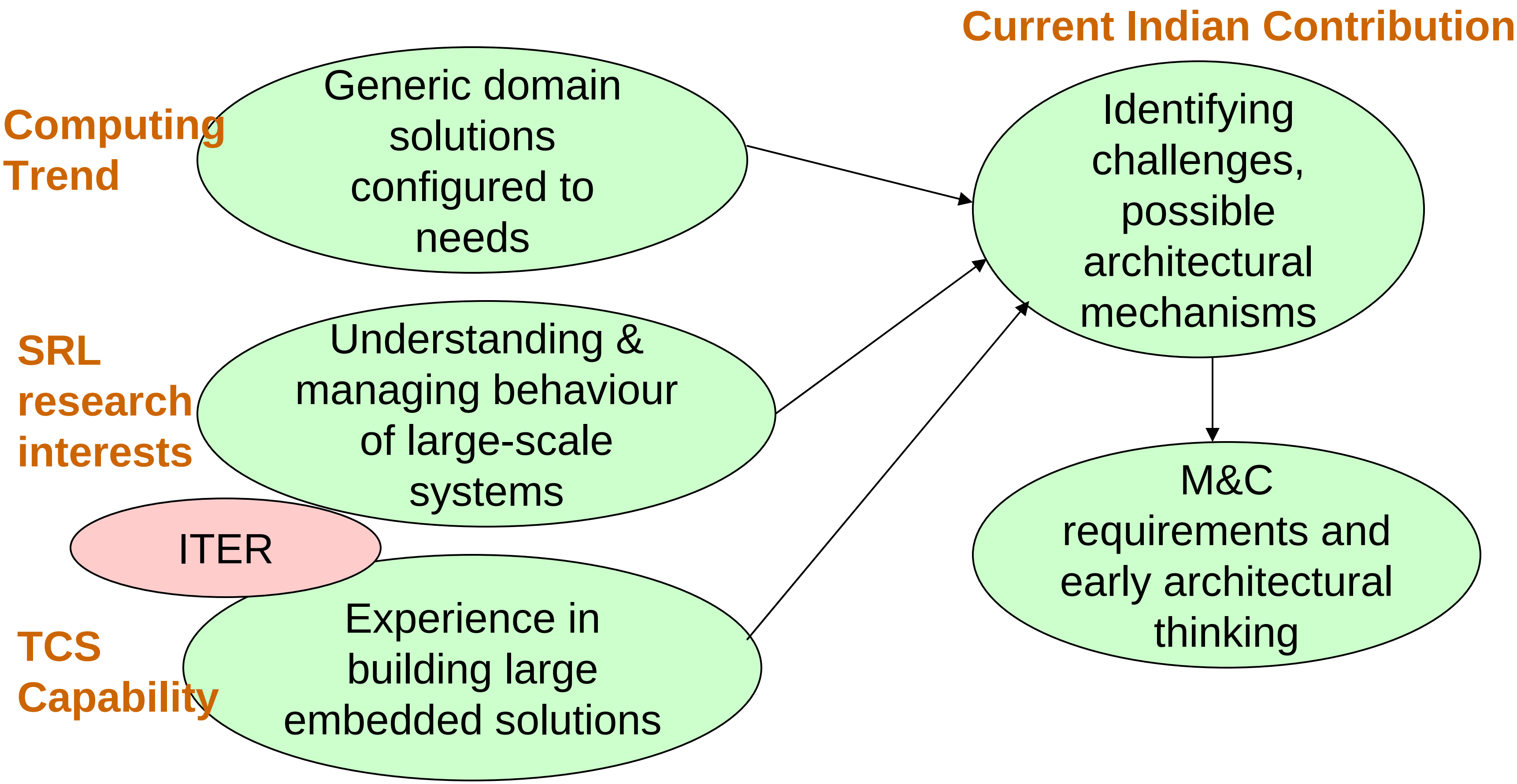
- Contribution Context
- Concept of Specification-driven Generic Architecture
- Scale challenges for M&C
  - Examples of generic architectural mechanisms to address them
- Next steps

Twin themes:

Challenges of scale

Approach to design: domain-generic architectural solutions

Context



Approach

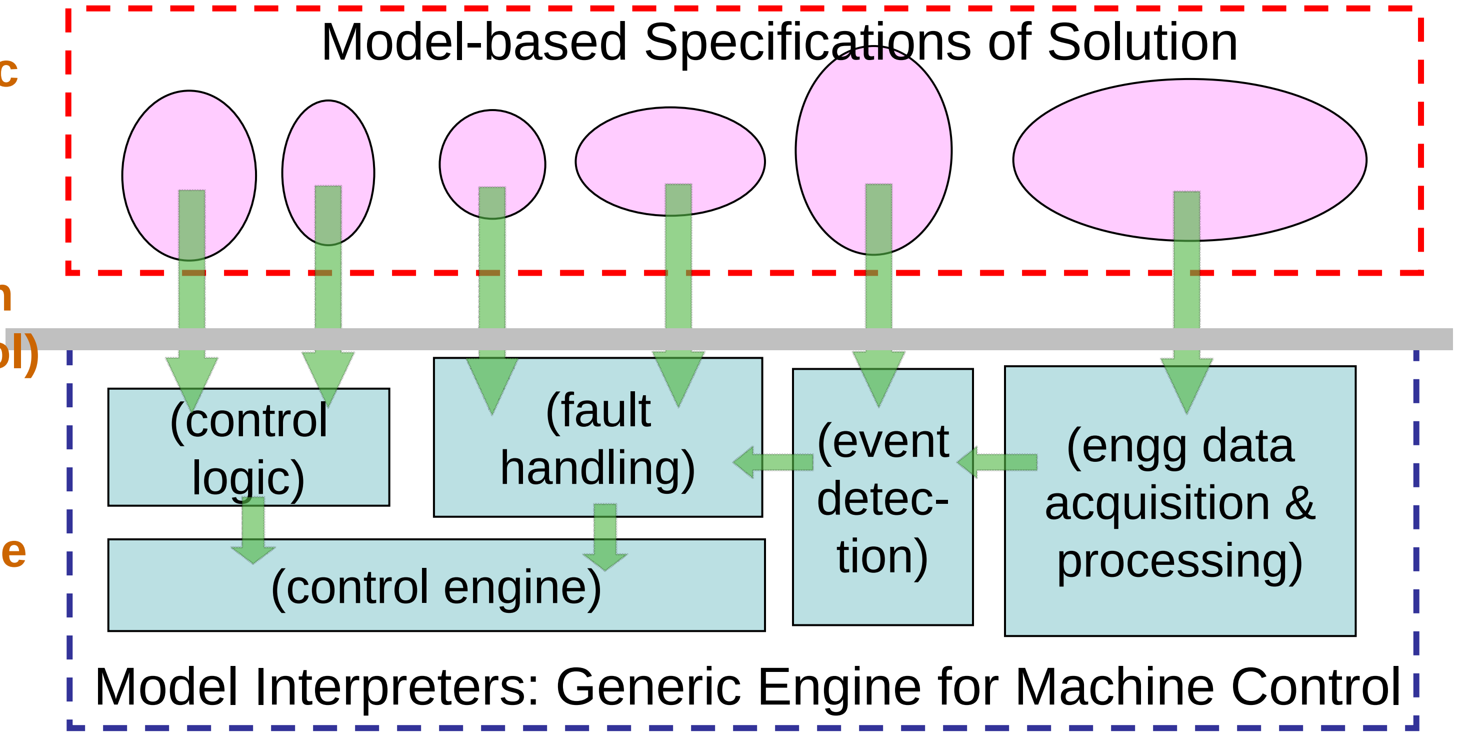
Identify and address system challenges through generic architectural mechanisms configurable to the specifics of the problem

Specification-driven (Model-driven) Architecture

Problem-specific Configuration Data

Model of domain (Machine Control)

Generic Software Architecture & Infrastructure



Model-driven software architectures are resilient to

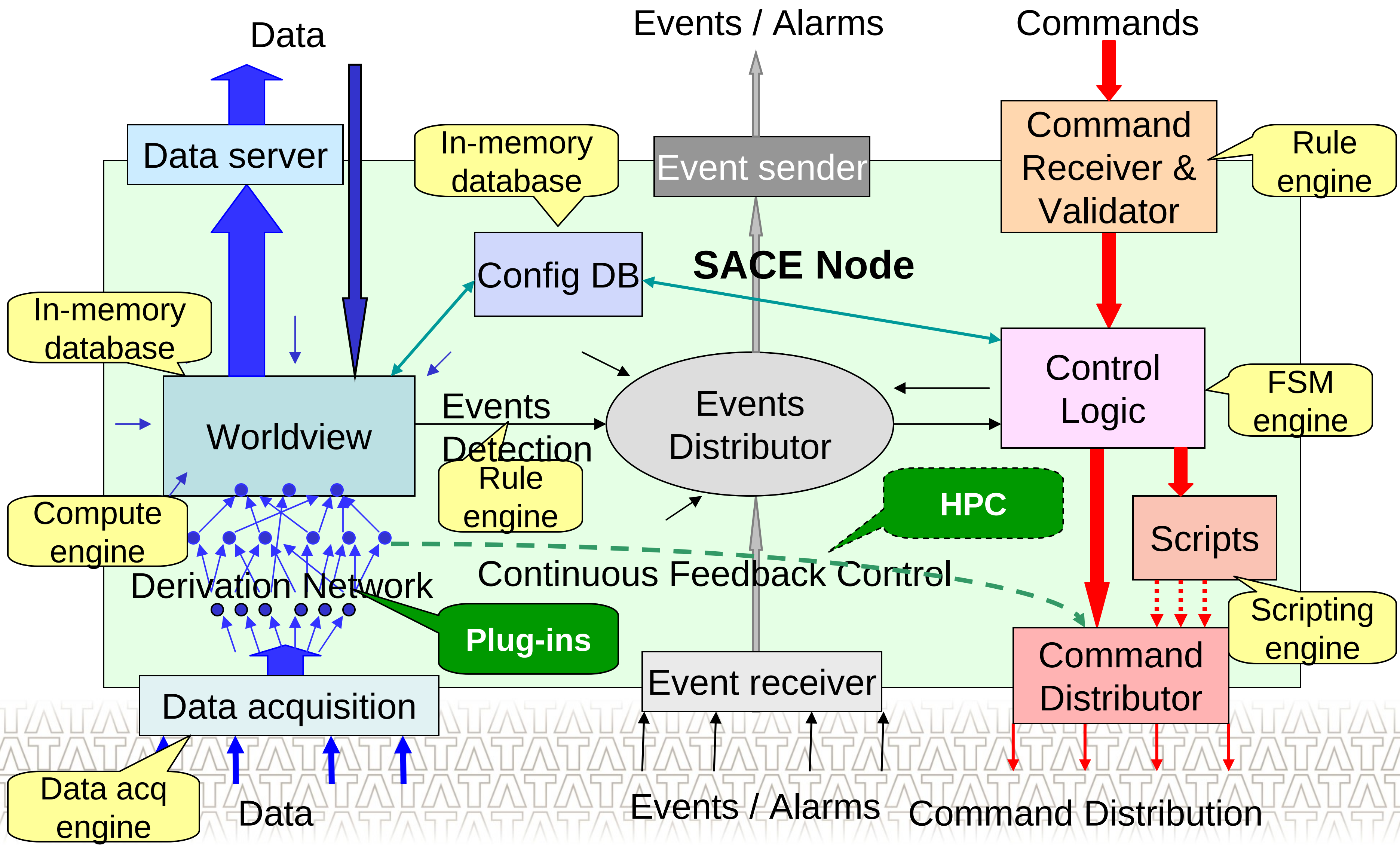
Changes in system architecture

Changes in realization technology

Sweet spot: Innovative approach that reduces risk!

Applies to all software, not just M&C

SACE Controller Node (open) Architecture



But how can system-specific architectural issues be addressed in a domain generic solution?

By abstracting them into generic architectural concerns applicable to some class of problems within the domain

We illustrate this for scalability concerns in M&C

Scale Challenges in M&C

Common challenges, transformed by scale

Operational Challenges

- Operator Monitoring and Management
- Automated Control
- Fault Detection & Handling

System Design & Management Challenges

- Engineering Properties
- System Behaviour Understanding and Management

Evolution Challenges

- Upgrades
- System Rollout & Evolution
- Software Evolution Costs & Agility

Operator Monitoring and Management

Abstract Requirements

- Multiple operators & users with different roles
  - Station operators, central operators, scientists possibly at remote locations
  - Possibilities of conflict
- Manual intervention infeasible on most alarms
- Must automate nearly all coordination & decision-making
  - Operator actions on groups of system elements

Scale issues

- Definable role capabilities, rules for avoiding / handling operator conflict
- Definable role-based monitoring dashboards with drill-down
  - Nearly no inline human responses to alarms

Generic Architectural Mechanisms

Automated Control and Coordination

Abstract Requirements

- Coordination of heterogenous groups with homogenous elements
  - Multiple technologies, sub-arrays
- Variations in control based on local factors
  - Support for automated situation handling

- Definable dynamic groups, definable control logic and scripting
- Principle of semi-autonomy: parent provides objectives, local choices on how to achieve objectives

Generic Architectural Mechanisms

Limitation: Redundant computations across homogenous elements

Optimization: Allow computation to "migrate" to parent

Fault Detection and Handling

Abstract Requirements

- Fully automated fault detection and handling
- Detection and handling of situations that affect groups of elements
- Faults with successively higher severity (e.g. dropping voltage)
  - Reponses to more severe faults must override previous responses
- Fault diagnosis across large numbers of elements and across time

Scale issue

- Definable fault detection and handling rules
- Hierarchical handling with awareness of fault relationships
- Multi-level fault detection and handling with different time horizons
  - Level 1 implements basic detection and reactive fault management
  - Level 2 detects patterns of faults and diagnosis of root causes
  - Level 3 detects trends across time for proactive system management

Generic Architectural Mechanisms

Engineering Properties

Abstract Requirements

- Achieve performance, reliability and other engineering properties at both element and system level
- Understand impact of configuration changes, faults and dynamic demands on system properties

Scale issue

- Create automated analysis tools to predict engineering properties
  - Dynamically calibrate the underlying system models to improve analysis
  - What-if analysis capabilities to guide system evolution
- Build in monitoring of engineering properties with feedback into the dynamic calibration of system models

Generic Architectural Mechanisms

System Behaviour Understanding and Management

Abstract Requirements

- Understand how the elements are behaving together as a system
  - Identify bottlenecks, opportunities to improve engineering properties e.g. power consumptions
- Detect and understand phenomena that affect groups of elements perhaps across time

Scale issues

- Automated offline data mining of engineering data (and quality properties of science data)
  - Build models of system behaviour
  - Identify problems and improvement opportunities

Generic Architectural Mechanisms

Data processing and visualization for engineering data!

Upgrades

Abstract Requirements

- Automation of software updates, support for system upgrades
- Challenge: May not be able to simultaneously evolve all elements due to local constraints
  - Interoperation and version management problems

Scale issue

- Architectural principle: parent version must be same as or newer than child version
  - Interoperation is the responsibility of the newer version
- Versioning mechanisms for data schema, control software

Generic Architectural Mechanisms

System Rollout and Evolution

Abstract Requirements

- Changes in coordination and fault handling logic to accommodate changes in system configuration
- Minimize changes to existing elements to accommodate new elements

Scale issues?

↓

- Definable control and fault-handling logic, facilitating evolution
- Principle of semi-autonomy: changes needed only to parent node of new element

Generic Architectural Mechanisms

System Evolution Cost and Agility

Abstract Requirements

- Cost and cycletime for accommodating system evolution
- Cost and cycletime for technology refresh

Scale issue?

- Specification-driven approach greatly reduces software maintenance cost and evolution cycletime
  - System evolution requires only changes to specifications (configuration data, plug-ins)
  - Technology refresh involves updating the generic engine, but not the specifications. Cost amortized across the domain.

Generic Architectural Mechanisms

Conclusions

- Computing field is moving away from custom solutions towards generic solutions for domains
  - Considerable benefits to SKA in adopting this approach
  - Applies beyond M&C to much of the software architecture
  - Critical parts of the data pipeline will still need custom development
- System scale poses challenges along multiple dimensions: system operation, system management and evolution
  - These challenges can be abstracted to the machine control domain and addressed with generic architectural mechanisms
  - Results in generic domain infrastructure, shared not only across radiotelescopes but many other scientific instruments such as ITER
- We would like to contribute to SKA based on these two areas of expertise: large-scale systems, model-driven architecture