Robotics is:

“the Intelligent Connection of Perception to Action”
Classical Robotic Control

“Pipelined” approach

Sensors → Perception → Modeling → Planning → Task Execution → Motor Control → Actuators
Practical Problems with Classical Control

- Slow to respond
- Perception
- Sensors
- Modelling
- Planning
- Task Execution
- Motor Control

Delayed reaction to stop robot

Slow to respond
Research Problems with Classical Control

Temptation to focus on model rather than reality

Other influences

Reality
Assumes a complete internal world model can be built, then manipulated

- Slow. Modelling must occur before robot can react to changes in environment
- Emphasis on using model is misleading-
  - Makes complex tasks seem solvable by directing attention away from perception
  - Makes potentially simple tasks complicated
- Requires crippling simplications
Alternatives to Classical Control

Environmental complexity vs. Cognitive complexity

- Traditional AI: complex cognition, simple environment
- Behaviour-based systems: simpler cognition, complex environment
- Reactive systems: simpler cognition, complex environment
Rather than functional decomposition, use “Behaviours”

- As much as possible, behaviours interact with each other through the environment, not the system
- The world is its own best model, so consult it directly whenever practical
- Use distributed representations tailored to the particular behaviours using them
Module $\text{act in parallel, in layers of increasing priority}$

Sensors
- All modules have access to sensors
  - Work usefully
  - Use maps
  - Explore
  - Wander
  - Avoid obstacles

Actuators
- All modules have access to actuators
  - Higher layers can view and modify the data flow in lower layers
Advantages of Subsumption

- Wander
- Explore
- Use maps
- Work usefully
- Avoidance
- Fast!

Reacts immediately to stop robot
Disadvantages of Subsumption

- Rigid priority scheme and strict layering are limiting -
  - Priorities must be evaluated and hardwired at design time
  - Requires behaviours to fit into a simple single-inheritance hierarchy
  - Behaviours cannot be combined, only enhanced linearly
Alternatives?

Traditional AI

- Environmental complexity
- Cognitive complexity

Extend scope of behaviour-based systems by improving behaviour “glue”

- Reactive systems
- Behaviour-based systems
"Lateral" Architecture

Priority flows between behaviours as they make use of each other

Work usefully
Use maps
Explore
Wander
Avoid obstacles

All modules have access to sensors
All modules have access to actuators

Sensors
Actuators
Lateral has a dynamic priority system designed to make it easy to build behaviours using other behaviours.

A behaviour is given the priority its highest priority user at a given time thinks it should have ("sponsorship").

Behaviours expose a limited public interface to users, rather than allowing access to their internal data flows.
Example Behaviour: Edge Following

EdgeM an

Capture → Compensate
Start ← Face
Waddle ← Stroll
Turn ←

iSide ← Logical Sensors

oNudge ← oMotor
Prowling Behaviour

ProwlMan

Prowling uses edge following in its implementation.
Prowling - Behaviours used

Prowling behaviour is built from a hierarchy of simpler behaviours.
ProwlMan active, and sponsoring edge following (EdgeMan)

NudgeMan beats SeekMan because, even though it is lower level, it has more sponsorship.
ProwlMan active, and sponsoring exploration (ExploreMan)
New behaviours fit in transparently to the ones already present.
Competition between behaviours is resolved by the “sponsor”
Extended syntax for expressing parallel processes (as state machines) and data flow between them.
Hardware: “Khepera®”

- Khepera miniature robot
- Processor: 68332
- Proximity sensors
- Light sensors
- Stepper motors
Priority Implementation: Connections

- Copy: A → A
- Compete/Subsume: B → A or B depending on priority
- Compete/Subsume: A, B or C depending on priority

A, B or C depending on priority