Automated Integration
Of Potentially Hazardous Open Systems

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, CA

Introduction

- A workshop talk is an opportunity for more speculative inquiry than usual. . .
- This talk is highly speculative!

An Anecdote

- A colleague who is an expert on certification is working with engineers building a ——— in ———
- The ——— engineers refuse to believe you have to do all this work for assurance and certification
  - We build it, test it, fix it, and it works
  - Then we have to spend 3 or 5 times that effort on safety assurance?
  - It's a ——— plot to hold us back
  - It cannot possibly require all this work
  - There must be a box somewhere that makes it safe
- I want to talk about that box!
  - The box that makes us safe

  In the context of open systems integration

Systems of Systems

- We're familiar with systems built from components
- But increasingly, we see systems built from other systems
  - Systems of Systems, SoS
- The component systems have their own purpose
  - Maybe at odds with what we want from them
- And generally have vastly more functionality than we require
  - Provides opportunities for unexpected behavior
  - Bugs, security exploits etc. (e.g., CarShark)
  - Emergent misbehavior
- Difficult when trustworthiness required
  - May need to wrap or otherwise restrict behavior of component systems
  - So, traditional integration requires bespoke engineering
  - Performed by humans

Self-Integrating Systems

- But we can imagine systems that recognize each other and spontaneously integrate
  - Possibly under the direction of an "integration app"
  - Examples on next several slides
- Furthermore, separate systems often interact through shared "plant" whether we want it or not (stigmergy)
  - e.g., separate medical devices attached to same patient

  And it would be best if they integrated "deliberately"
- These systems need to "self integrate"
  - Speculate system evolution can be framed in same terms
- And we want the resulting system to be trustworthy
- Which may require further customization of behavior
- And construction of an integrated assurance case

Scenarios

- I'll describe some scenarios, mostly from medicine
- And most from Dr. Julian Goldman (Mass General)
  - "Operating Room of the Future" and
  - "Intensive Care Unit of the Future"
- There is Medical Device Plug and Play (MDPnP) that enables basic interaction between medical devices
- And the larger concept of "Fog Computing" to provide reliable, scaleable infrastructure for integration
- But I'm concerned with what the systems do together rather than the mechanics of their interaction

Anesthesia and Laser

- Patient under general anesthesia is generally provided enriched oxygen supply
- Some throat surgeries use a laser
- In presence of enriched oxygen, laser causes burning, even fire
  - A new hazard not present in either system individually
- So, want laser and anesthesia m/c to recognize each other
- Laser requests reduced oxygen from anesthesia machine
- But. . .
  - Need to be sure laser is talking to anesthesia machine connected to this patient
  - Other (or faulty) devices should not be able to do this
  - Laser should light only if oxygen really is reduced
  - In emergency, need to enrich oxygen should override laser

Other Examples

- I'll skip the rest in the interests of time
- But they are in the slides (marked SKIP)

Heart-Lung Machine and X-ray SKIP

- Very ill patients may be on a heart-lung machine while undergoing surgery
- Sometimes an X-ray is required during the procedure
- Surgeons turn off the heart-lung machine so the patient's chest is still while the X-ray is taken
- Must then remember to turn it back on
- Would like heart-lung and X-ray mc's to recognize each other
- X-ray requests heart-lung machine to stop for a while
  - Other (or faulty) devices should not be able to do this
  - Need a guarantee that the heart-lung restarts
- Better: heart lung machine informs X-ray of nulls

Patient Controlled Analgesia and Pulse Oximeter SKIP

- Machine for Patient Controlled Analgesia (PCA) administers pain-killing drug on demand
  - Patient presses a button
  - Built-in (parameterized) model sets limit to prevent overdose
  - Limits are conservative, so may prevent adequate relief
- A Pulse Oximeter (PO) can be used as an overdose warning
- Would like PCA and PO to recognize each other
- PCA then uses PO data rather than built-in model
- But that supposes PCA design anticipated this
- Standard PCA might be enhanced by an app that manipulates its model thresholds based on PO data
- But. . .

PCA and Pulse Oximeter (ctd.) SKIP

- Need to be sure PCA and PO are connected to same patient
- Need to cope with faults in either system and in communications
  - E.g., if the app works by blocking button presses when an approaching overdose is indicated, then loss of communication could remove the safety function
  - If, on the other hand, it must approve each button press, then loss of communication may affect pain relief but not safety
  - In both cases, it is necessary to be sure that faults in the blocking or approval mechanism cannot generate spurious button presses
- This is hazard analysis and mitigation at integration time

Blood Pressure and Bed Height SKIP

- Accurate blood pressure sensors can be inserted into intravenous (IV) fluid supply
- Reading needs correction for the difference in height between the sensor and the patient
- Sensor height can be standardized by the IV pole
- Some hospital beds have height sensor
  - Fairly crude device to assist nurses
- Can imagine an ICU where these data are available on the local network
- Then integrated by monitoring and alerting services
- But. . .

Blood Pressure and Bed Height (ctd.) SKIP

- Need to be sure bed height and blood pressure readings are from same patient
- Needs to be an ontology that distinguishes height-corrected and uncorrected readings
- Noise- and fault-characteristics of bed height sensor mean that alerts should be driven from changes in uncorrected reading
- Or, since, bed height seldom changes, could synthesize a noise- and fault-masking wrapper for this value
- Again, hazard analysis and mitigation at integration time

What's the Problem?

- Since they were not designed for it
- It's unlikely the systems fit together perfectly
- So will need shims, wrappers, adapters, monitors etc.
- So part of the problem is the "self" in self integration
- How are these customizations constructed automatically during self integration?

What's the Problem? (ctd. 1)

- In many cases the resulting assembly needs to be trustworthy
  - Preferably do what was wanted
  - Definitely do no harm

- Even if self-integrated applications seem harmless at first, will often get used for critical purposes as users gain (misplaced) confidence
  - E.g., my Chromecast setup for viewing photos
  - Can imagine surgeons using something similar
    (they used Excel!)

- So how do we ensure trustworthiness, automatically?

Models At Runtime (M@RT)

- If systems are to adapt to each other
- And wrappers and monitors are to be built at integration-time
- Then the systems need to know something about each other
- One way is to exchange models
  - Machine-processable (i.e., formal) description of some aspects of behavior, claims, assumptions
- This is Models at RunTime: M@RT
- When you add aspects of the assurance case, get Safety Models at RunTime: SM@RT (Trapp and Schneider)
- Most recent in a line of system integration concepts
  - Open Systems, Open Adaptive Systems, System Oriented Architecture

Four Levels of SM@RT

Due to Trapp and Schneider, but this is my version

1. Unconditionally safe integration
   - The component systems guarantee safety individually, with no assumptions on their environment
   - It follows that when two or more such systems are integrated into a SoS, result is also unconditionally safe
2. Conditionally safe integration
   - The component systems guarantee safety individually, but do have assumptions on their environment
   - When two such systems are integrated into a SoS, each becomes part of the environment of the other
   - It is necessary for them to exchange their models and assurance arguments and to prove that the assumptions of each are satisfied by the properties of the other
   - The resulting system will also be conditionally safe

Four Levels of SM@RT (ctd. 1)

3. Safely managed integration
   - This class is similar to the previous one except the component systems are not able to ensure each others' assumptions
   - Hence one or both systems must be customized in some way, generally by synthesizing a wrapper or runtime monitor that excludes the troublesome cases
   - For example, if one system delivers an unacceptable result, a runtime monitor/enforcer can block it and signal failure to the other system
   - Or if one system cannot deliver the assumed behavior in some cases, a wrapper can block or transform its inputs to exclude those cases

Four Levels of SM@RT (ctd. 2)

4. Safe integration despite hazards
   - In this class, it is possible that the integrated system has new hazards (i.e., potentially unsafe circumstances) not present with either system individually
   - For example, a surgical laser may be safe and an anesthesia machine may be safe, but the combination possesses a new hazard that the laser can cause burning and fire in the enriched oxygen supplied by the anesthesia machine
   - Once the hazards are known, this class can be transformed into the previous one (e.g., the laser can be disabled if the anesthesia machine is delivering enriched oxygen, or the anesthesia machine can be instructed not to use enriched oxygen if the laser is operating)

SM@RT Examples

- I think DEOS does SM@RT levels 1, 2, maybe 3, but probably not fully automatically
- Mario Trapp et al at Frauenhofer do level 2 for John Deere (tractors and agricultural implements)
- Semantic Interoperability Logical Framework SILF is Level 3
  - Developed by NATO to enable dependable machine-to-machine information exchanges among Command and Control Systems
  - Extensive ontology to describe content of messages
    - ⋆ So in SM@RT terms, ontological descriptions (e.g., in OWL) are the models
  - Mediation mechanism to translate messages as needed
    - ⋆ Synthesized at integration time
  - ONISTT is an SRI prototype of these capabilities, now a spinoff

SM@RT Automation

- SM@RT substitutes automation at integration time for human activities performed at design time
- Furthermore, these activities are traditionally thought to require significant human expertise
  - Verification, customization, hazard analysis
- However, each of these can be thought of, and organized as, a search over model(s)

  Verification: automated by mechanized deduction (SAT, SMT, and quantifiers), which is pure search

  Customization: a form of synthesis, which can be organized as a further search on top of mechanized deduction
  - Guess a solution (can be guided by templates)
  - Try to verify its correctness
  - If that fails use counterexamples to help refine the guess and iterate

SM@RT Automation (ctd.)

Hazard Analysis: it's also a search, but over a vast space of possibilities

- Not just computational interactions but all kinds
- Generally requires human "greybeards" who mentally sweep the space of possibilities to find the significant ones, rather like a master chess or go player, without (apparently) doing explicit search
- Even greybeards miss things, so there are systematic processes such as HAZOP and STPA
- HAZOP uses abstracted models and asks "what happens if this value is —?" where — is selected from a catalog of "guidewords" such as "missing," "late," "small," etc.

Human Expertise vs. Search

- What makes automated verification and synthesis successful is the quality of the models over which the search is performed
  - And sustained improvement in how to do the search
- Search does not replace human expertise
- Instead the expertise shifts from doing the activity itself to building the models that enable the activity to be automated by search
- I speculate that it is now feasible to build models that support hazard analysis
- Should have multiple models, each representing a different point of view

Automating Hazard Analysis

- The models of the greybeards are often highly abstract
  - Boxes and arrows
- Previously infeasible to compute over these
- But can now do it (INF-BMC over uninterpreted functions)
- Need models of component systems, and of their environment
- Let's start with something fairly constrained
  - e.g., medical devices
  - Environment is human physiology and surrounding plant
- This model could be a community-wide resource
  - Whole industry, regulators, public, could cooperate on its development and validation
  - Might not be correct at first: new incidents and accidents will be factored in and won't happen again (cf. Tesla)

Hazard Models, Trivial Sketch

- Environment model contains an element saying that a source of energy in conjunction with a "large" a flow of oxygen triggers a potential "burn" or "fire" hazard
- The model of a laser notes that it is a source of energy
- Model of an anesthesia machine records the possibility that it can produce enhanced (i.e., "large") oxygen
- Then search will reveal the potential "burn" hazard in the composed SoS
- Hazards for fire likely independent of those for overdose, so compositional
- Hence, feasible. . . maybe

Another Anecdote

- Microsoft's "Tay" was a Twitter bot that the company described as an experiment in "conversational understanding"
  - The more you chat with Tay, said Microsoft, the smarter it gets, learning to engage people through "casual and playful conversation"
  - Within less than a day of its release, it had been trained by a cadre of bad actors to behave as a racist mouthpiece and had to be shut down

- Lots of things are "put out there" without thought for the potential hazards of their interaction with the world at large

- What if we could anticipate these unfortunate interations?

Future Vision

- Some years from now. . .
- Can imagine a community-developed hazard model for "the world at large"
- When deploying new systems, do Level 4 integration against this model
- Model acts as a surrogate for the world
- The world is its own implementation, but its model resides in a computational system (a box) to which new systems connect and integrate
- Hazard analysis and customizations then ensure safe integration with the world at large
- It's the box that makes us safe!

Summary

- We are moving to a world where human-constructed designs are surpassed by those derived from human-constructed sketches explored and optimized by automated search
- Human skill and expertise retain—even increase—their value, but it is expressed in sketches and models rather than individual designs
- That creates opportunities where useful artifacts can be constructed automatically by search on generic models
- Safe and dependable integration of open systems could be one of the first realizations of these capabilities
- The SM@RT hierarchy suggests a road map for development
  - Level 1 is here, 2 is achievable, 3 is feasible
- And I want to suggest that 4, automated hazard analysis, is at least conceivable
- And would be a social good