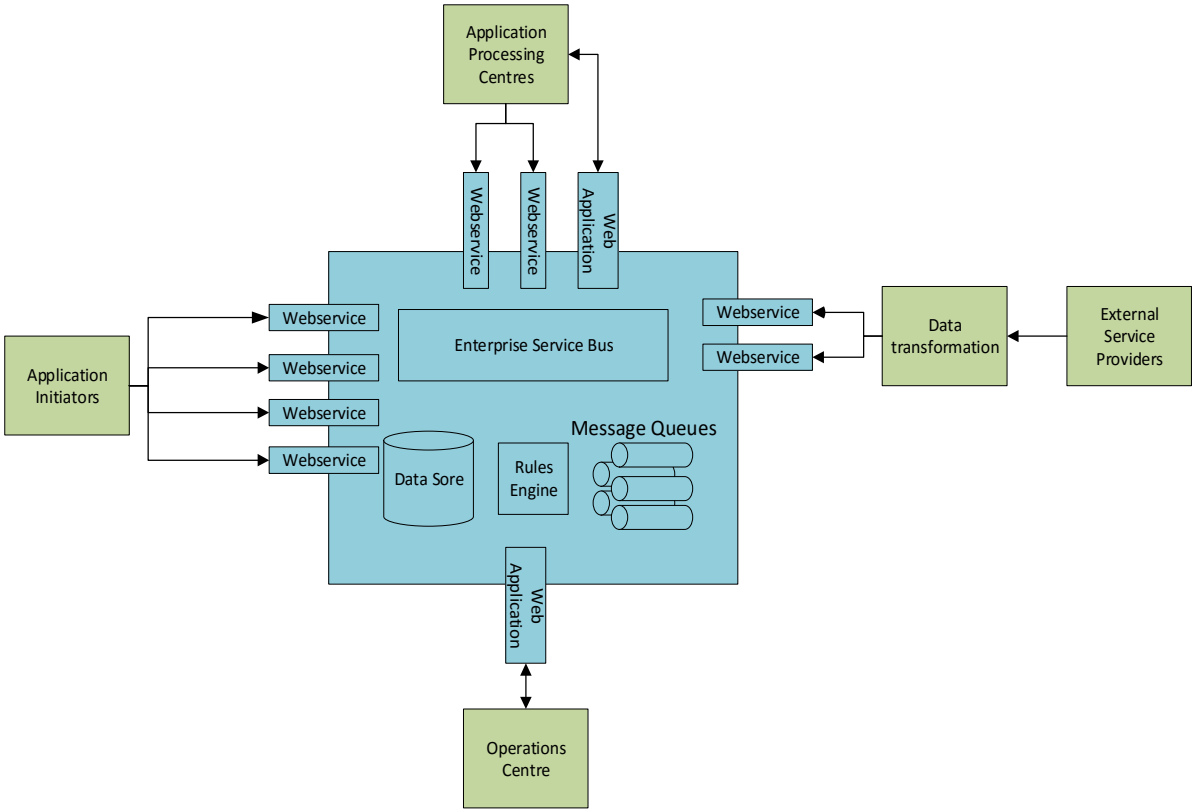
Testing in the Age of Complexity

BILL MATTHEWS

@BILL_MATTHEWS

A recent *complicated* project



- Event Driven Processing of Applications
- Complicated business rules (~1.3M unique input combinations)
- Complicated interdependences between systems

- However the system is still:
  - Deterministic - Linear cause and effect
  - Static System Behaviours

What if…

…the systems we test:

- Learned their behaviours from an environment rather than using prescribed behaviours?
- Automatically adapted its behaviours over time to better fit with the operating context?
- Generated outcomes that were not deterministic and might be wrong some of the time?

…our systems:

- Consisted of networks of such learning and adaptive systems?
- Were designed to directly compete or collaborate with each other?

We are already seeing (and using) such systems

- The technology is becoming more accessible now – lower barrier to entry

How would this be different to testing our current systems?

Do we ideas or intuitions on what might be important to test in such systems?

A Trivial Example – Algorithms gone weird

I've regularly bought products from Amazon since 1999

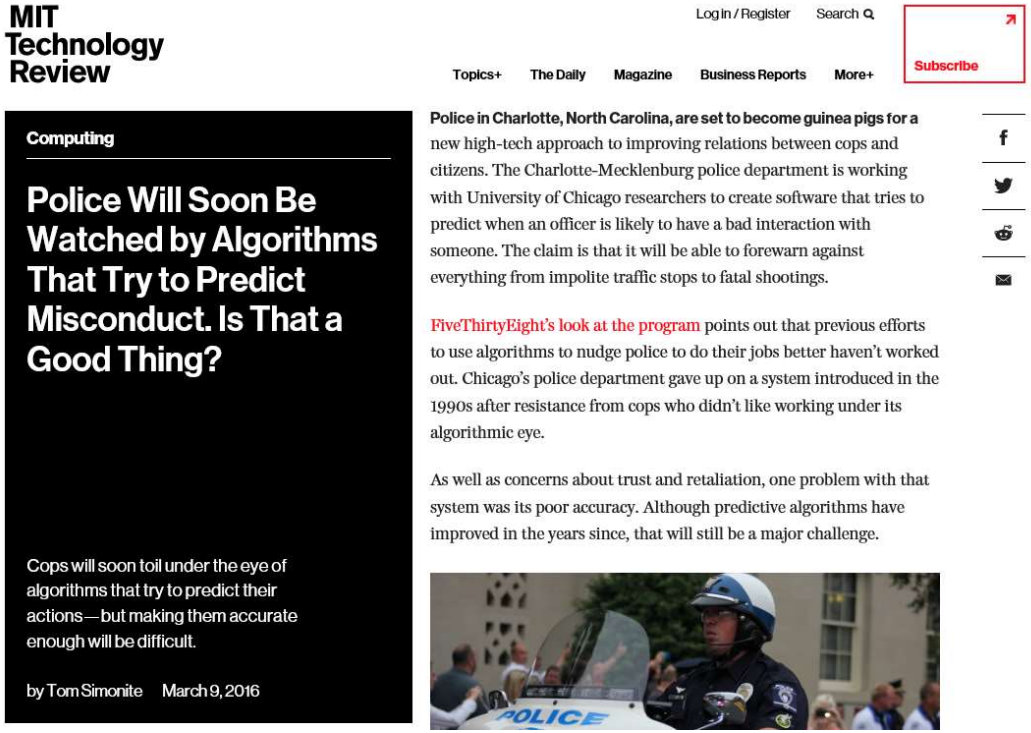
- The have a pretty good idea of what I like to buy

One Christmas, I bought my wife a handbag. Afterwards...

- They thought I'd want more handbags
- They also thought I might like catfood

They mostly get it right but this time they lost an opportunity to sell me something – big deal

A more serious example – what could go wrong here?



MIT Technology Review

Log In / Register Search

Topics+ The Daily Magazine Business Reports More+

Subscribe

Computing

Police Will Soon Be Watched by Algorithms That Try to Predict Misconduct. Is That a Good Thing?

Cops will soon toil under the eye of algorithms that try to predict their actions—but making them accurate enough will be difficult.

by Tom Simonite March 9, 2016

Police in Charlotte, North Carolina, are set to become guinea pigs for a new high-tech approach to improving relations between cops and citizens. The Charlotte-Mecklenburg police department is working with University of Chicago researchers to create software that tries to predict when an officer is likely to have a bad interaction with someone. The claim is that it will be able to forewarn against everything from impolite traffic stops to fatal shootings.

FiveThirtyEight's look at the program points out that previous efforts to use algorithms to nudge police to do their jobs better haven't worked out. Chicago's police department gave up on a system introduced in the 1990s after resistance from cops who didn't like working under its algorithmic eye.

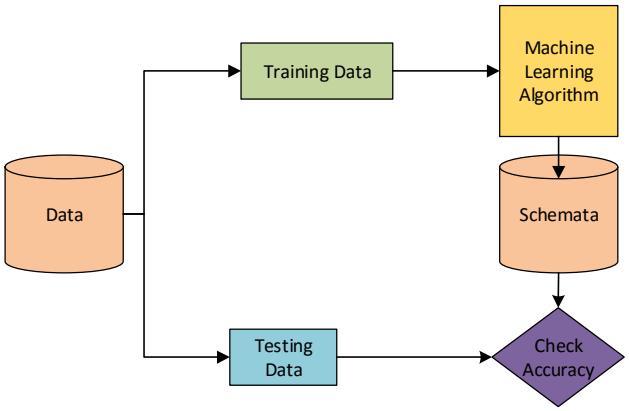
As well as concerns about trust and retaliation, one problem with that system was its poor accuracy. Although predictive algorithms have improved in the years since, that will still be a major challenge.

Something more fun to think about

http://how-old.net/#

https://www.what-dog.net/#

How does it *Learn?*



A simplified generic Machine Learning Process

- Available data is split into *Training* and *Testing* sets
- Algorithm *learns* based on the *Training* data
- The *Schemata* is a representation of the learned knowledge.
  - This may not make sense to a human observer
  - This is almost certainly different to the Schemata a human might form
- Accuracy of the derived *Schema* is based on the previously unseen *Testing* data.
- Usually an iterative process to find the *best* mix of input features, type of algorithm and algorithm tuning parameters.
- *Best Performing* combination becomes the Schemata we use.
  - *In general it's never 100% accurate*
  - *In some contexts accuracy of 75% might be OK.*

Does this model of the system help us?

What biases might be at work during learning and our testing of the learning?

Some ideas to consider

The *Schemata* learned by the machine

- Is biased by the data used to train and assess accuracy
  - A lack of **Requisite Variety** compared to the operating context will result in failures regardless of the machine's performance.
- Is almost certainly very different to that learned by a human
- We may not full understand what has actually been *learned*

We generally value the behaviour that aligns with our own *Schemata*

- We give human-like Agency to machines where a high degree of correlation between the two is evident.

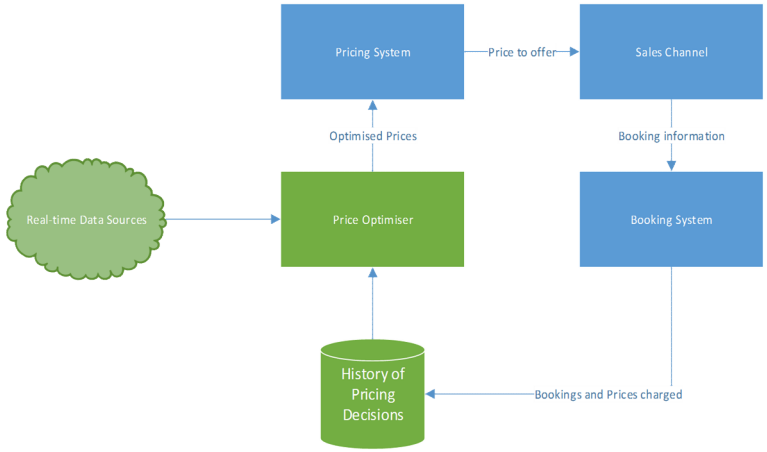
The system behaviour maybe be incorrect but since the system is not 100% accurate – how do we determine if it's a problem or not?

Learning machines can fail in ways that a human would not

- As Testers we need to consider how the learning may fail in ways that matter to our stakeholders.
- Think about testing as exploring the machine's Schemata to formulate

Additional Reading:
Requisite Variety: https://en.wikipedia.org/wiki/Variety_(cybernetics)

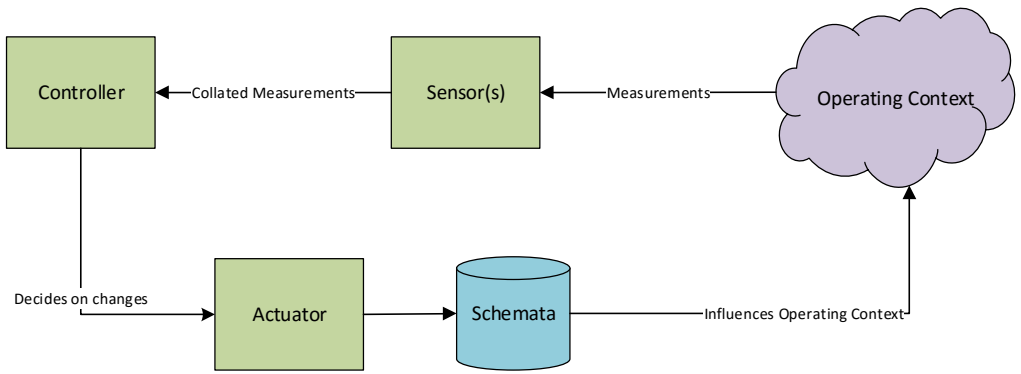
Adaptive Pricing Project



- System aims to optimise utilisation by manipulating prices to influence customer behaviour
  - Higher prices *usually* dis-incentivise one purchase over another
  - Lower prices *usually* incentivise one purchase over another
- Overall Pricing Model may change daily.
  - A price for a product today may not yield the same price tomorrow for the same conditions
- Prices are set by an algorithm that adapts to the context over time.

"Is this going to loose us revenue?" – question asked by a Senior Stakeholder.
What strategies could we apply to provide insights into this question?
Would replaying the purchases for the last month or year answer this question?

How do such systems adapt?



- We collect some measurements (*Sensors*) from our *Operating Context*.
- We analyse the measurements against our goals and decide on changes (*Controller*)
- We enact the changes in our *Schemata* (*Actuator*)
- Our new *Schemata* (hopefully) influences the *Operating Context.*
- Adaptation may reach some form of steady (static or oscillating) state over time (equilibrium)

Does this model of the system help us?
How might the system adapt in ways that are undesirable?
What experiments will help us understand how the system adapts/evolves?

Some ideas to consider

We are unlikely to be able to provide **THE ANSWER**
- If we could we would probably not use an adaptive system
- We may be able to provide a compelling testing story that provides insights

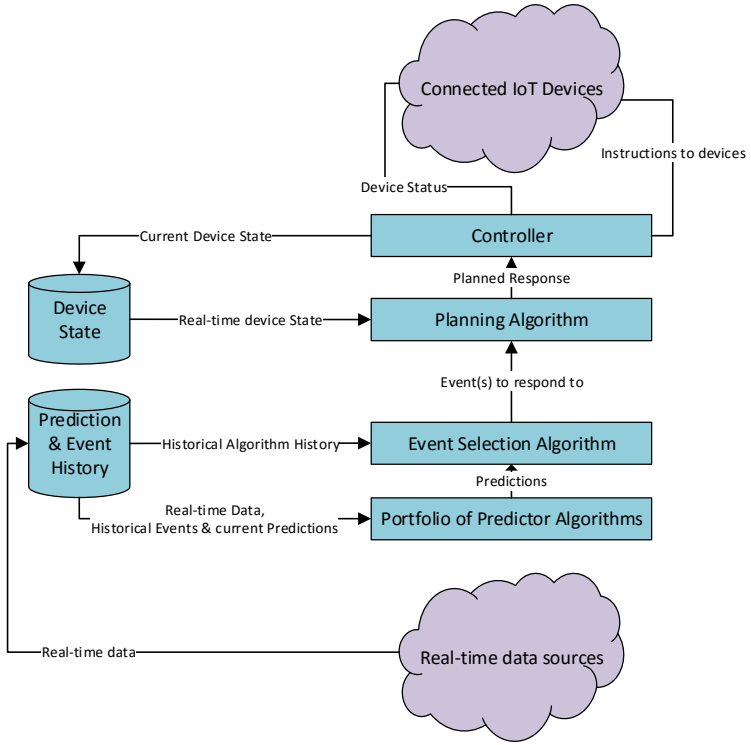
Replaying historic data may help (or parallel running in live or Test in Production)
- So long as past behaviour is a good indicator of future behaviour

Our system is designed to influence behaviour of other Agents (customers)
- Historical data may not be a good indicator of future behaviour under influence
- We may be able to model the anticipated range of behaviours and run simulations based on historical patterns.
- The models and result of simulations will help us present the testing story

Not all adaptation is beneficial - see https://en.wikipedia.org/wiki/Tay_(bot) for an example
- Do we have Oracles that help us identify there is a problem or if we are trending towards one?

Some changes in the environment may cause a greater change in our system
- What future scenarios may occur that are not covered by the historical data?
- Is the system sensitive to particular changes? Does this matter?

Complexity – another recent project



This System aims to:

- **Predict** the occurrence of a range of important/critical events
  - Predictions are time-series of probabilities of events occurring that are updated in real-time.
- **Select** the events to respond to
  - May not be able to respond to all events
  - Selection adapts based on accuracy of predictions made and success of actions.
- **Plan** a response to critical events using a large estate of connected devices (IoT)
  - May not be able to create a plan for a given event
  - Needs to consider utilisation of IoT devices
- **Monitor** and **Adapt** the plan in response to the outcomes of our actions, feedback from IoT devices and any new events.

How do we test this?
Will testing each component/layer in isolation be sufficient?

Ideas to consider

The previous ideas about testing *learning systems* and *adapting systems* are still relevant here

In complex systems (adaptive or not)…

- Behaviours may emerge that we cannot foresee from the parts
  - Testing the components (e.g. the Controller) in isolation is important but we may miss emergent behaviours that are considered important.
  - Isolating and testing feedback loops may be more appropriate than components
  - Isolating and testing externalities may also be appropriate
- We may not be able to predict *explicit expected behaviours* in complex systems (or within feedback loops)
  - But we need to be able to recognise potential problem or undesirable behaviours – Heuristics

Topics I'm currently thinking about

Larger Networks of Connected Learning and Adapting Systems

- Previous examples have focused on a specific narrow activity
- How would we think about testing a *Smart City/Town/Building?* These consist of a wider range of different connected systems and devices that:
  - Form wider & longer dependency chains
  - Form wider & longer feedback loops
  - Form wider and longer externalities

Learning and Adaptive Systems that

- Need to collaborate with other systems (that we have limited knowledge of)
- Are in competition with other systems (that we have limited knowledge of)

Malicious Intent

- How might a system be coerced into evolving unwanted behaviour? How would we test for this?

Closing Thoughts

Increase your "*Complexity Threshold*" – your capacity to think critically about complex systems.

- Practice designing test strategies for complex systems
  - Don't be limited by your current projects (e.g. how would you test the "Police Misconduct Prediction Algorithm" shown earlier?). Try something like that as a team exercise and capture your ideas
  - Don't be overly concerned about having the *right* answer – just exploring the idea of the system will raise your threshold
  - Be a student of topics related to complex systems such as *General Systems Thinking, Cybernetics, Complexity Theory, Network Theory, Complex Adaptive Systems Theory, Simulation, Statistics* and *Probability.*
    - There are ideas and models in these areas that will be helpful when dealing with complexity
    - **Go as deep as is useful to you** – don't get bogged down by these topics – graze them

Think critically about your test strategies and tests

- Why is this test important?
- What do I expect to get out of my testing?
- What information is my testing not providing me that might be important?
- How might my testing be fooling me?