SEVENTH FRAMEWORK PROGRAMME

MIDAS
testing on cloud

RASEN
Compositional Risk Assessment and Security Testing of Networked Systems

Effizientere IT-Sicherheitstests mit Hilfe von Usage-based Testing

GI TAV 37
5. Februar 2015

Martin Schneider
Fraunhofer FOKUS

Steffen Herbold
Universität Göttingen

Outline

- **Challenge: Efficiency of Security Testing**
- **Introduction to Fuzzing**
  - Data Fuzzing
  - Behavioral Fuzzing
- **Introduction to Usage-Based Testing**
- **FORTHCOMING ACTIVITIES**

Challenge: Efficiency of Security Testing

- **Security testing, e.g. fuzzing, may be very complex, dependent on**
  - data types, input space
  - fuzzing techniques

1600 complex data types

$$\mathcal{O}\left(\sum_{i=1}^{n} \frac{o'!}{(o'-i)!}\right)$$

- **Brad Arkin (CSO of Adobe) at RSA Conference 2013:**
  - *„Fuzzer are the perfect solution for consultants. When started fuzzing, they will never go again. This type of test never comes to an end.“*

INTRODUCTION TO SECURITY TESTING

Introduction to Fuzzing: Definition

- **Fuzzing is about injecting invalid or random inputs**
  - to obtain **unexpected behavior**
  - to identify **errors** and potential **vulnerabilities**
- **Interface robustness testing**
- **Fuzzing is able to find (zero-day-)vulnerabilities, e.g.**
  - crashes
  - denial of service
  - security exposures
  - performance degradation
- **No false positives**

Introduction to Fuzzing: Categorization

dumb

smart

- **Random-based fuzzers** generate randomly input data. They don't know nearly anything about the SUT's protocol.

  fuzzed input:HdmxH&k dd#**&%

- **Template-based fuzzers** uses existing traces (files, …) and fuzzes some data.

  template: **GET /index.html**

  fuzzed input:**GE? /index.html, GET /inde?.html**

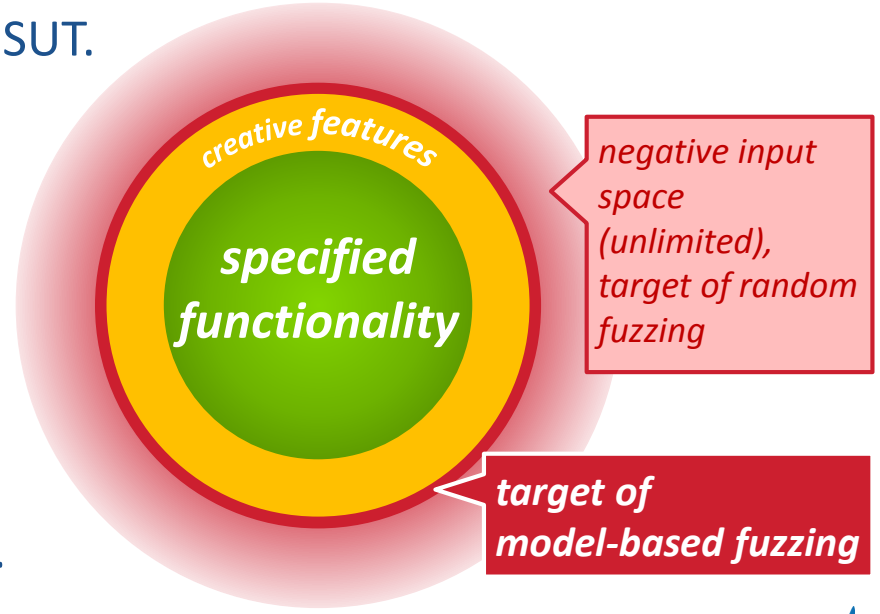
- **Block-based fuzzers** break individual protocol messages down in static (grey) and variable (white) parts and fuzz only the variable part.

  fuzzed input:**GET /inde?.html, GET /index.&%ml**

- **Dynamic Generation/Evolution-based fuzzers** learn the protocol of the SUT from feeding the SUT with data and interpreting its responses, for example using evolutionary algorithms.

Introduction to Fuzzing: Model-based Fuzzer

- **Model-based fuzzers** uses models of the input domain (protocol models, e.g. context free grammars), for generating systematic non-random test cases
- The model is executed on- or offline to generate complex interaction with the SUT.
- Thus it is possible fuzz data after passing a particular point.
- Model-based fuzzers finds defects which human testers would fail to find.



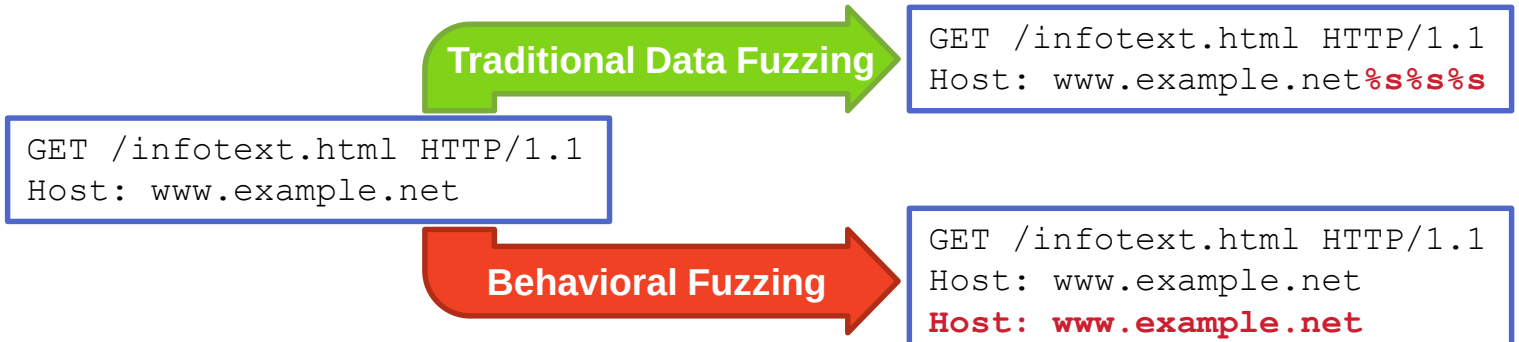
see also: Takanen, A., DeMott, J., Miller, C.: Fuzzing for Software Security Testing and Quality Assurance. Artech House, Boston (2008)

Introduction to Fuzzing: Behavioral Fuzzing

- Traditional fuzzing generates only invalid input data to find vulnerabilities in the SUT.
  Behavioral fuzzing complements traditional fuzzing by not fuzzing only input data of messages but **changing the appearance and order of messages**, too.

- The **motivation** for the idea of fuzzing behavior is that vulnerabilities cannot only be revealed when invalid input data is accepted and processed but also when invalid sequences of messages are accepted and processed.

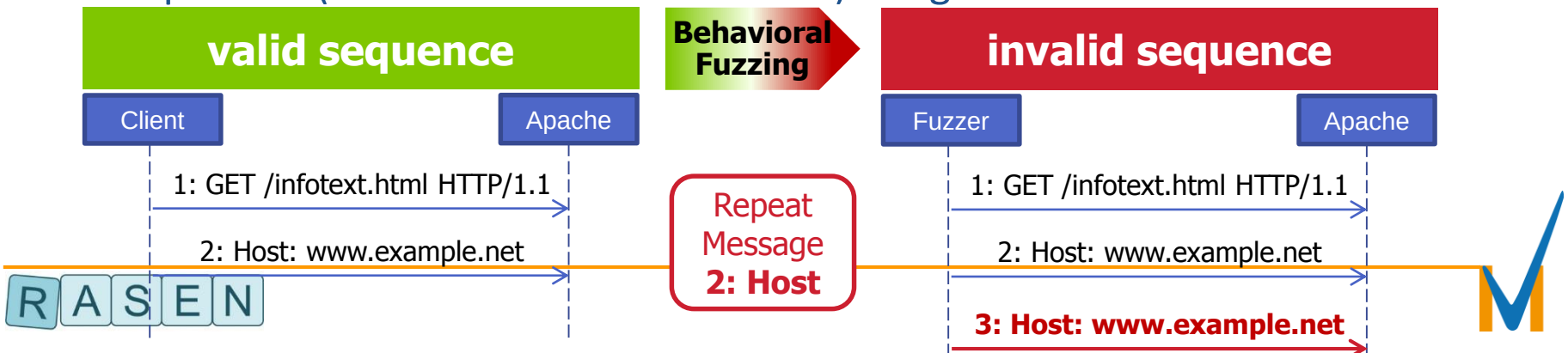
- A real-world example is given in [KHK10] where a vulnerability in Apache web server was found by repeating the host header message in an HTTP request.



[KHK10] Kitagawa, T., Hanaoka, M., Kono, K.: AspFuzz: A State-aware Protocol Fuzzer based on Application-layer Protocols. In: IEEE Symposium on Computers and Communications, pp.202-208 (2010)

Behavioral Fuzzing of UML Sequence Diagrams

- Test cases are generated by **fuzzing one or more valid sequences**.
- This concrete fuzzing of behavior is realized by changing the order and appearance of messages in two ways:
  - **By rearranging messages directly**. This enables straight-lined sequences to be fuzzed.
    Fuzzing operators are for example remove, move or repeat a message.
  - **By utilizing control structures of UML 2.x sequence diagrams**, such as combined fragments, guards, constraints and invariants. This allows more sophisticated behavioral fuzzing that avoids less efficient random fuzzing.
- By applying one ore more fuzzing operators to a valid sequence, invalid sequences (= behavioral fuzz test cases) are generated.



INTRODUCTION TO USAGE-BASED TESTING

Event-driven Software is Everywhere

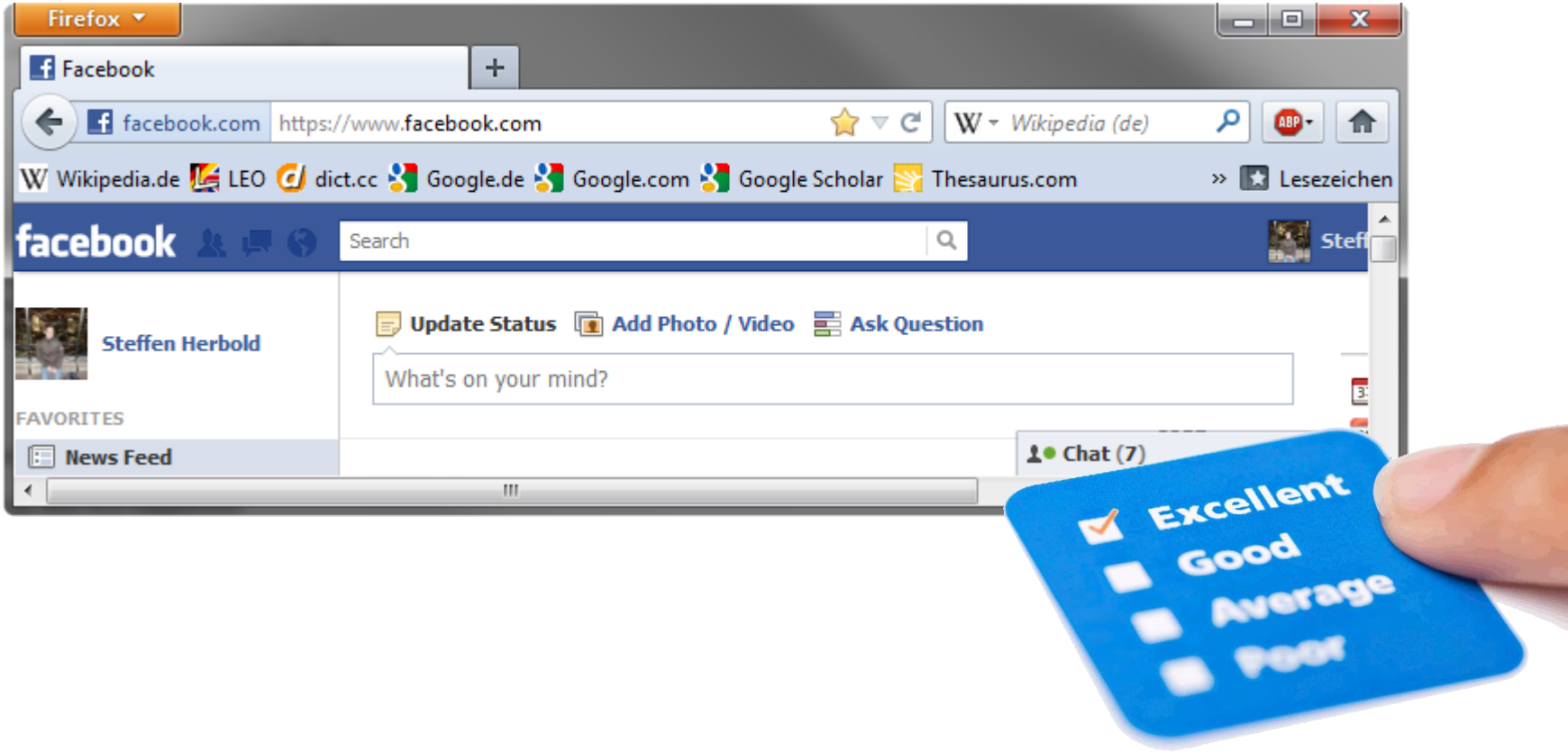


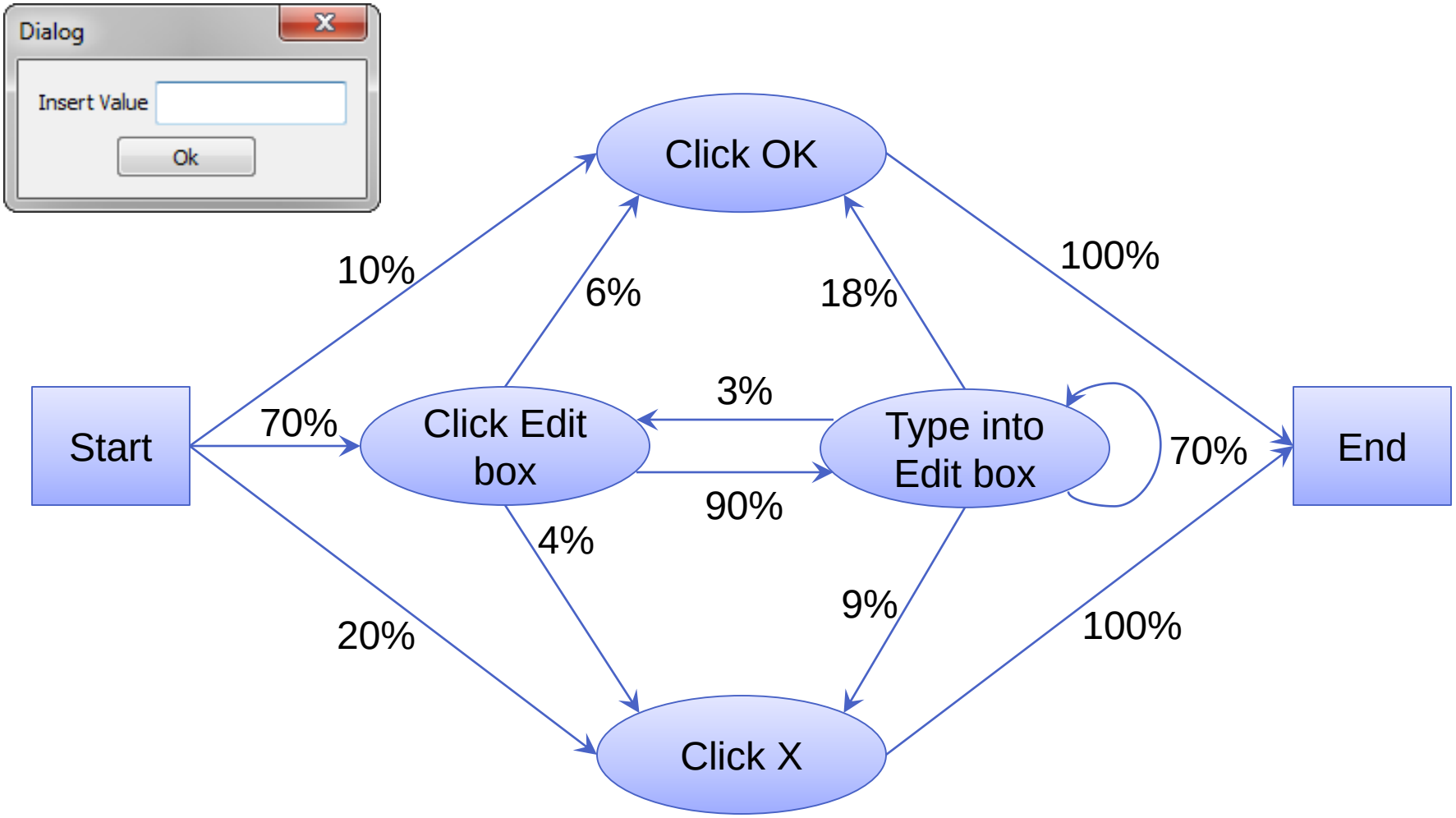
Image source: http://www.brentwoodconstruction.ie/quality-assurance#/quality-assurance/

Usage-based Testing

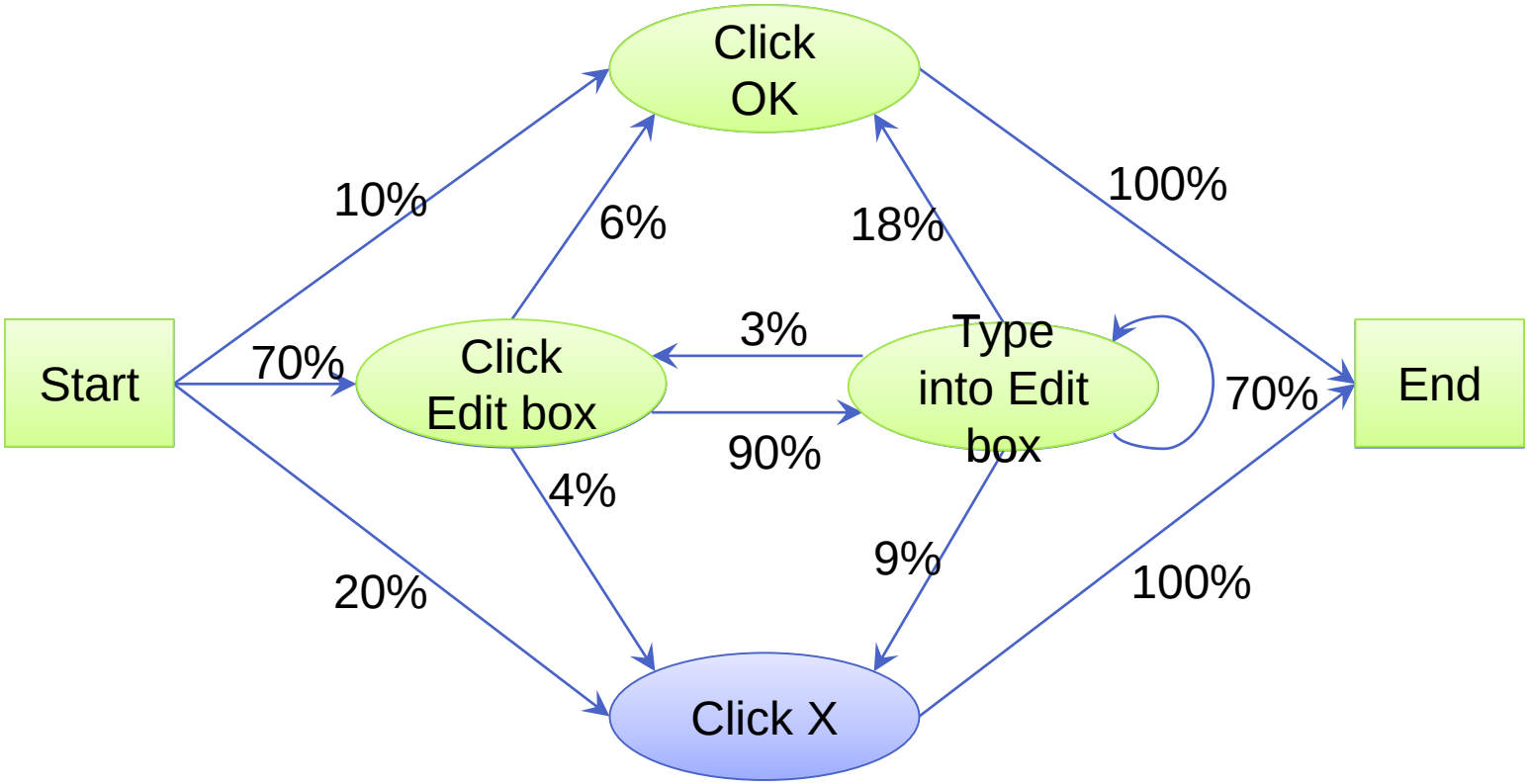
- **Prioritizes efforts based on the software's usage**
  - Not every function is equally important

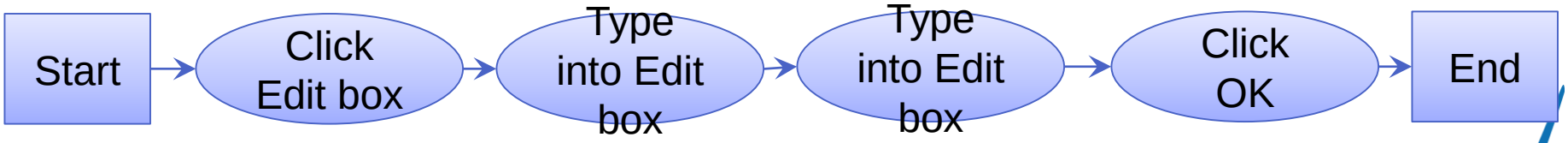
- **Usage profiles model behavior as stochastic process**

Events and Usage Profiles



Random Walk


Click
OK
10%
6%
18%
100%
Start
70%
Click
Edit box
3%
90%
Type
into Edit
box
70%
End
4%
9%
20%
100%
Click X


Start
Click
Edit box
Type
into Edit
box
Type
into Edit
box
Click
OK
End

USAGE-BASED FUZZING

Challenge: Efficiency of Security Testing

- **Security testing, e.g. fuzzing, may be very complex, dependent on**
  - data types, input space
  - fuzzing techniques

1600 complex data types

$$\mathcal{O}\left(\sum_{i=1}^{n} \frac{o'!}{(o'-i)!}\right)$$

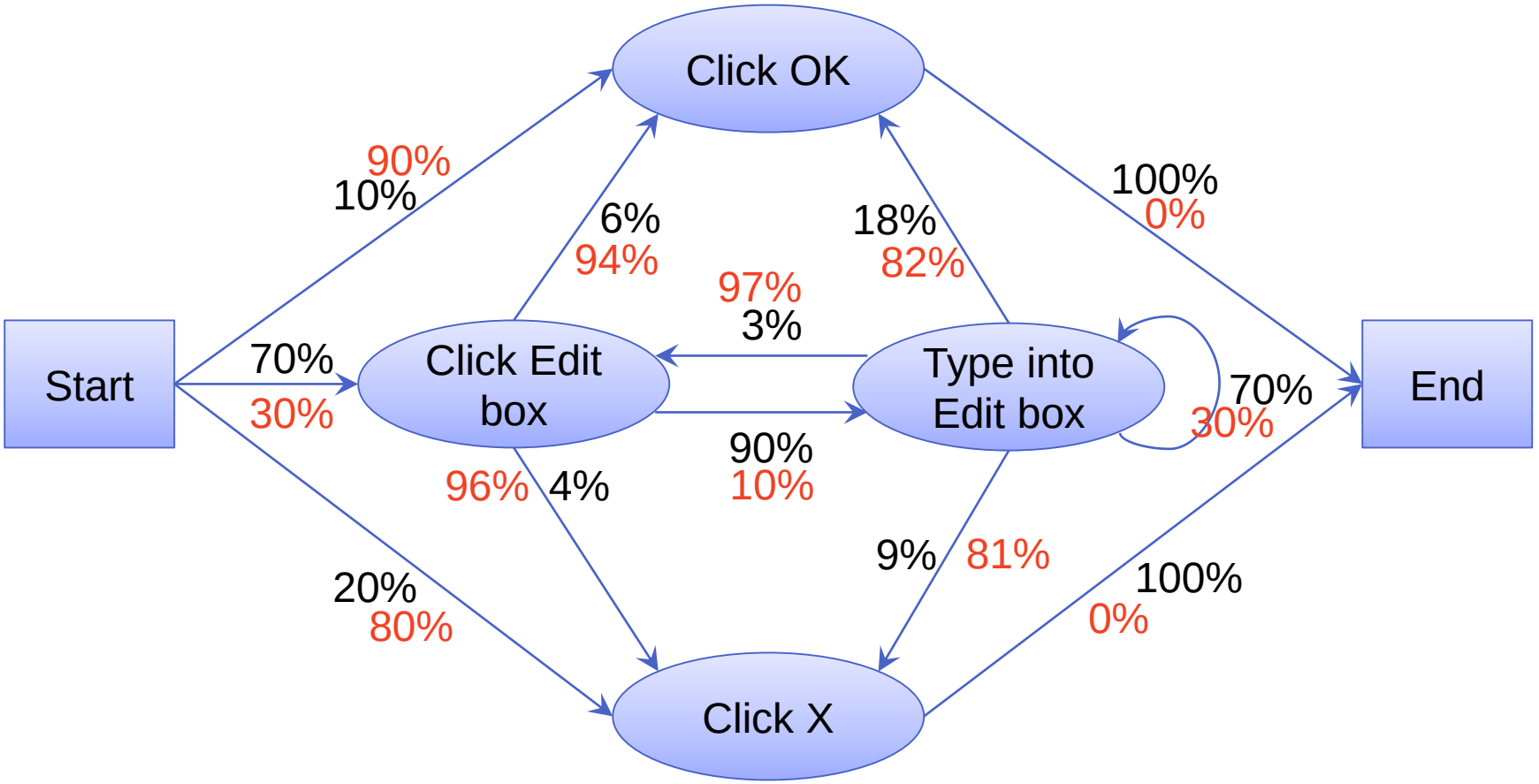
- **Brad Arkin (CSO of Adobe) at RSA Conference 2013:**
  - *„Fuzzer are the perfect solution for consultants. When started fuzzing, they will never go again. This type of test never comes to an end.“*

Basic Idea

- **An attacker is not interested in the usage frequency of a function when perfoming an attack**
  - all functions may become target of an attack with the same probability
- **Assuming that less tested functionality is more error prone, including security relevant bugs**
  - successful attacks are more likely on rarely tested functionality
- **Usage profiles can be used to identify rarely used/tested functionality**

Adaptation of Usage-Based Profile

- **Invert probabilities of transitions**
  - serves as input for generation of complex test scenarios

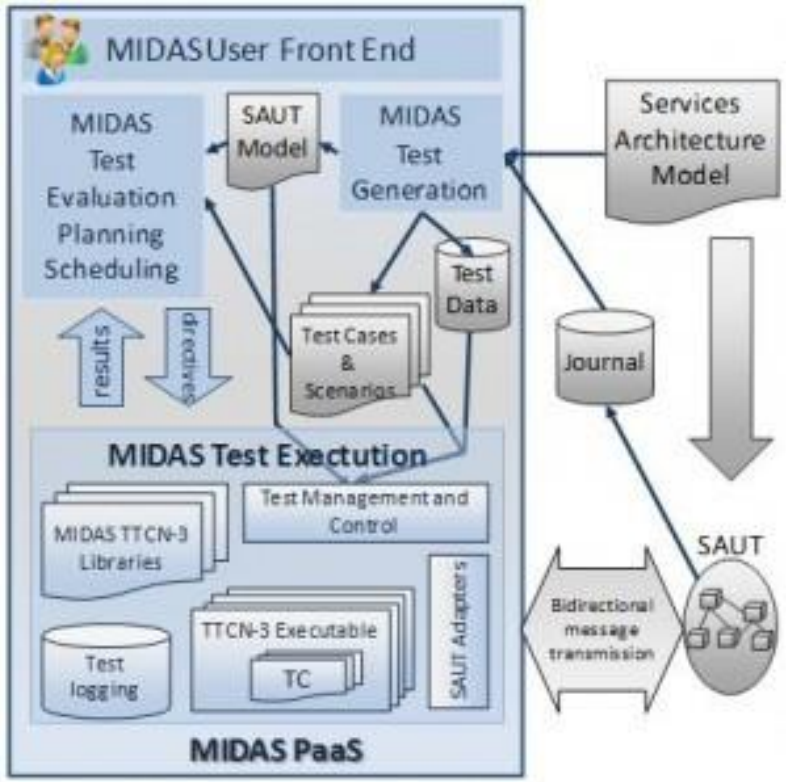


Adaptation of Usage-Based Profile

- **test scenarios generated from the inverted usage profile serve as**
  - starting point for behavioral fuzzing
    - *seldom used message sequences and transitions*
  - starting point for data fuzzing
    - *usage profile also contains input data*
    - *the less data the higher the probality to apply data fuzzing*

Next Steps: Evaluation on Use Cases of MIDAS Project

- **The goal of the MIDAS project is to design and build a test automation facility that targets SOA implementations**



- **Test methods are implemented as services**
  - usage-based tesing and security testing can be orchestrated
- **Application and evaluation on case studies from**
  - Logistics domain
  - Healthcare domain

SEVENTH FRAMEWORK PROGRAMME

MIDAS testing on cloud

RASEN Compositional Risk Assessment and Security Testing of Networked Systems

Thank you for your attention.

Any questions or remarks?