Can we compose software with building blocks and verify it automatically?

Francis Leung

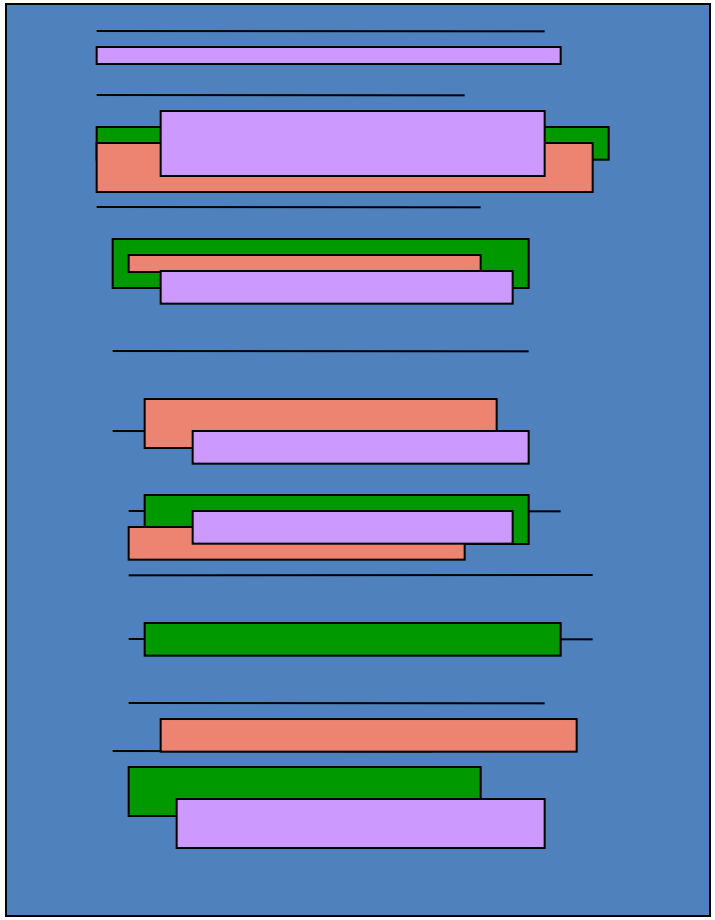
Outline

- Why we work on these two problems
- Challenges and our approach
- What we are working on

Two Problems in Software Engineering

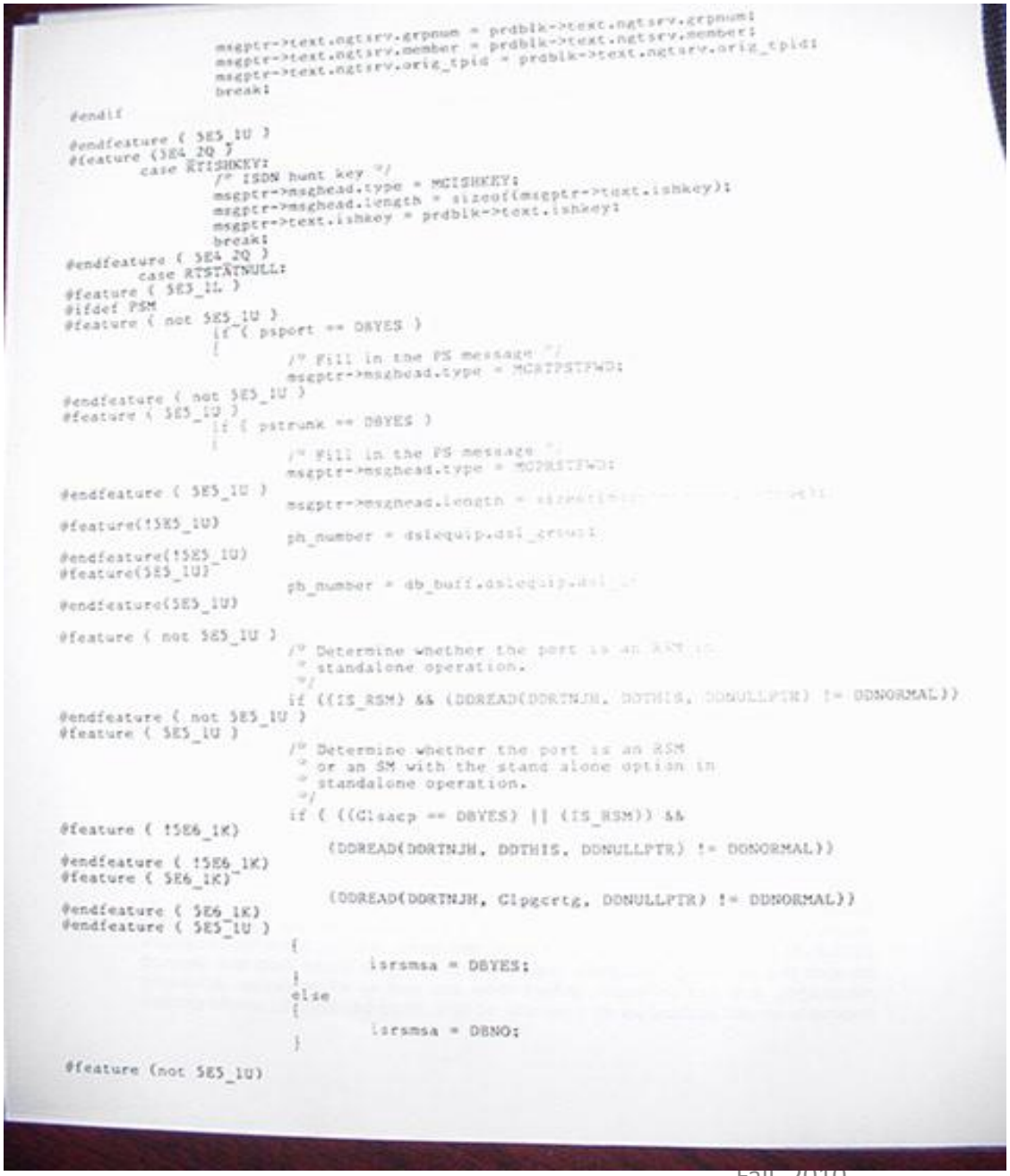
- Almost all programmers in the industry make their living by changing code and in testing
- Software maintenance cost is 60% to 90% of the total software development cost
  - Can we add features without changing code?
- Software testing often takes half the time and resources of a software project
  - Can we verify software by assertions automatically?

What It Takes to Add Features to SW Today



POTS as An Example

- The programmer develops **Call Waiting** by changing the code of **POTS**
- **Call Forwarding** is developed by changing the code of **Call Waiting** and **POTS**
- **Retry** is developed by changing the code of **Call Forwarding**, **Call Waiting** and **POTS**, and so on
- The programmer is not just implementing one feature. He must thoroughly understand change the code and test many features
- Features are not reusable without each other
  - They are "**entangled**"



```
                msgptr->text.ngtsrv.grpnum = prdblk->text.ngtsrv.grpnum;
                msgptr->text.ngtsrv.member = prdblk->text.ngtsrv.member;
                msgptr->text.ngtsrv.orig_tpid = prdblk->text.ngtsrv.orig_tpid;
                break;
#endif

#endfeature ( 5E5_1U )
#feature (5E4_2Q )
        case RTISHKEY:
                /* ISDN hunt key */
                msgptr->msghead.type = MCISHKEY;
                msgptr->msghead.length = sizeof(msgptr->text.ishkey);
                msgptr->text.ishkey = prdblk->text.ishkey;
                break;
#endfeature ( 5E4_2Q )
        case RTSTATNULL:
#feature ( 5E3_1L )
#ifdef PSM
#feature ( not 5E5_1U )
                if ( psport == DBYES )
                {
                        /* Fill in the PS message */
                        msgptr->msghead.type = MCRTPSTFWD;
#endfeature ( not 5E5_1U )
#feature ( 5E5_1U )
                if ( pstrunk == DBYES )
                {
                        /* Fill in the PS message */
                        msgptr->msghead.type = MCRTSTFWD;
#endfeature ( 5E5_1U )
                        msgptr->msghead.length = sizeof(
#feature(!5E5_1U)
                        ph_number = dslequip.dsl_circuit
#endfeature(!5E5_1U)
#feature(5E5_1U)
                        ph_number = db_buff.dslequip.dsl_c
#endfeature(5E5_1U)

#feature ( not 5E5_1U )
                        /* Determine whether the port is an RSM (S
                         * standalone operation.
                         */
                        if ((IS_RSM) && (DDREAD(DDRTNJH, DDTHIS, DDNULLPTR) != DDNORMAL))
#endfeature ( not 5E5_1U )
#feature ( 5E5_1U )
                        /* Determine whether the port is an RSM
                         * or an SM with the stand aloone option in
                         * standalone operation.
                         */
                        if ( ((Clsacp == DBYES) || (IS_RSM)) &&
#feature ( !5E6_1K)
                            (DDREAD(DDRTNJH, DDTHIS, DDNULLPTR) != DDNORMAL))
#endfeature ( !5E6_1K)
#feature ( 5E6_1K)
                            (DDREAD(DDRTNJH, Clpgcrtg, DDNULLPTR) != DDNORMAL))
#endfeature ( 5E6_1K)
#endfeature ( 5E5_1U )
                        {
                                lsrsmsa = DBYES;
                        }
                        else
                        {
                                lsrsmsa = DBNO;
                        }
#feature (not 5E5_1U)
```

How does entangled code look like?

Why Are We Still Writing Entangled Code?

The programmer is not experienced

The programmer is not well trained

The programmer is not given enough time and resources.

Because they use top-down

No, because they use bottom-up

It is due to object oriented programming

It is because they do not follow a plan (agile programming)

It is due to water fall

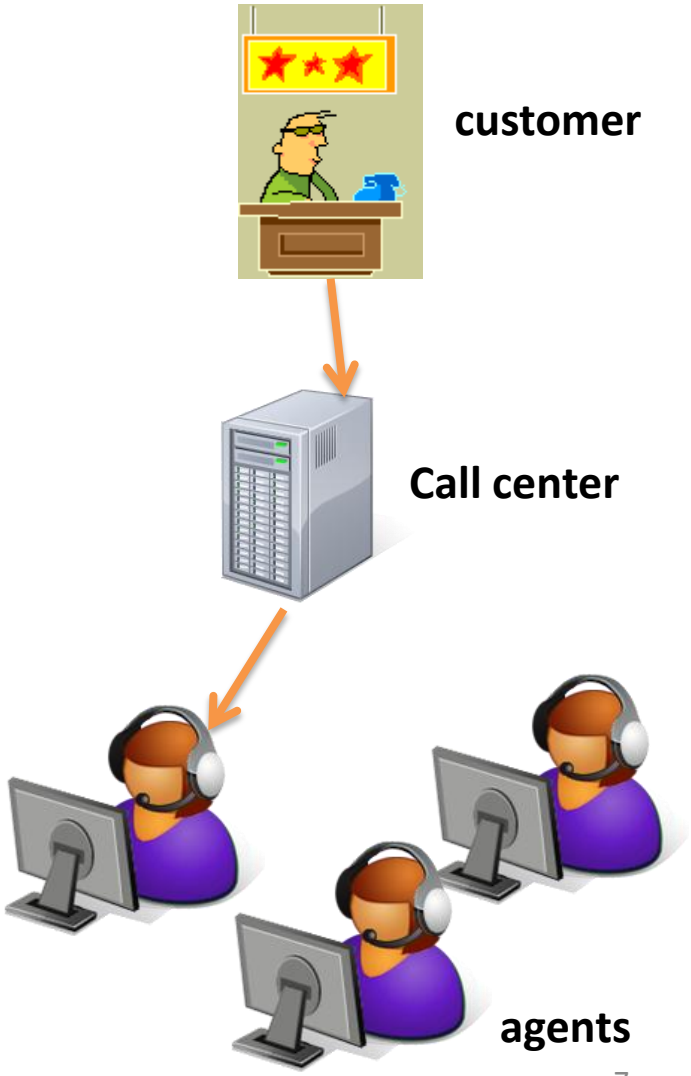
"Whoever blames will later be blamed. The times they are a changing" – I. Bob Dylan

The programmers cannot help it!

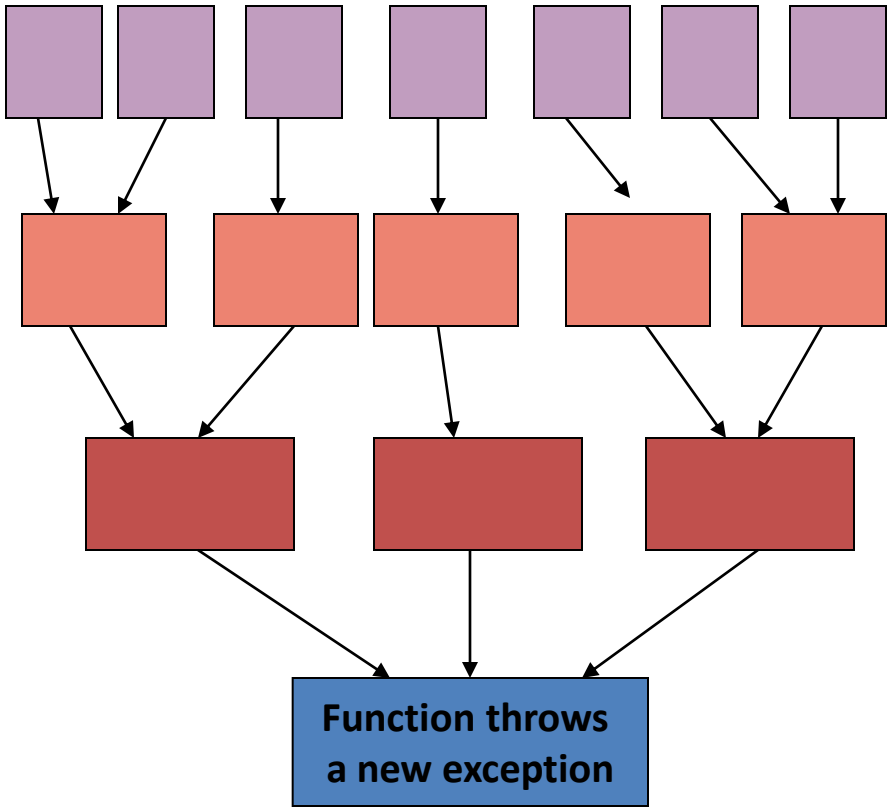
W. H. Leung, "Program entanglement, feature interaction and the Feature Language Extensions," Computer Networks, Feb., 2007

Why do we keep on changing code?

```
try {  ……
          if ( agent.isAvailable() )
          call.transferTo (agent);
          …..
}
catch (AgentCrashed) {
          // Transfer call to another agent
          ……
}
catch (NetworkCongestion) {
          // Do other things
          ……
} catch (NetworkDown) {
          // Do yet other things
}
```

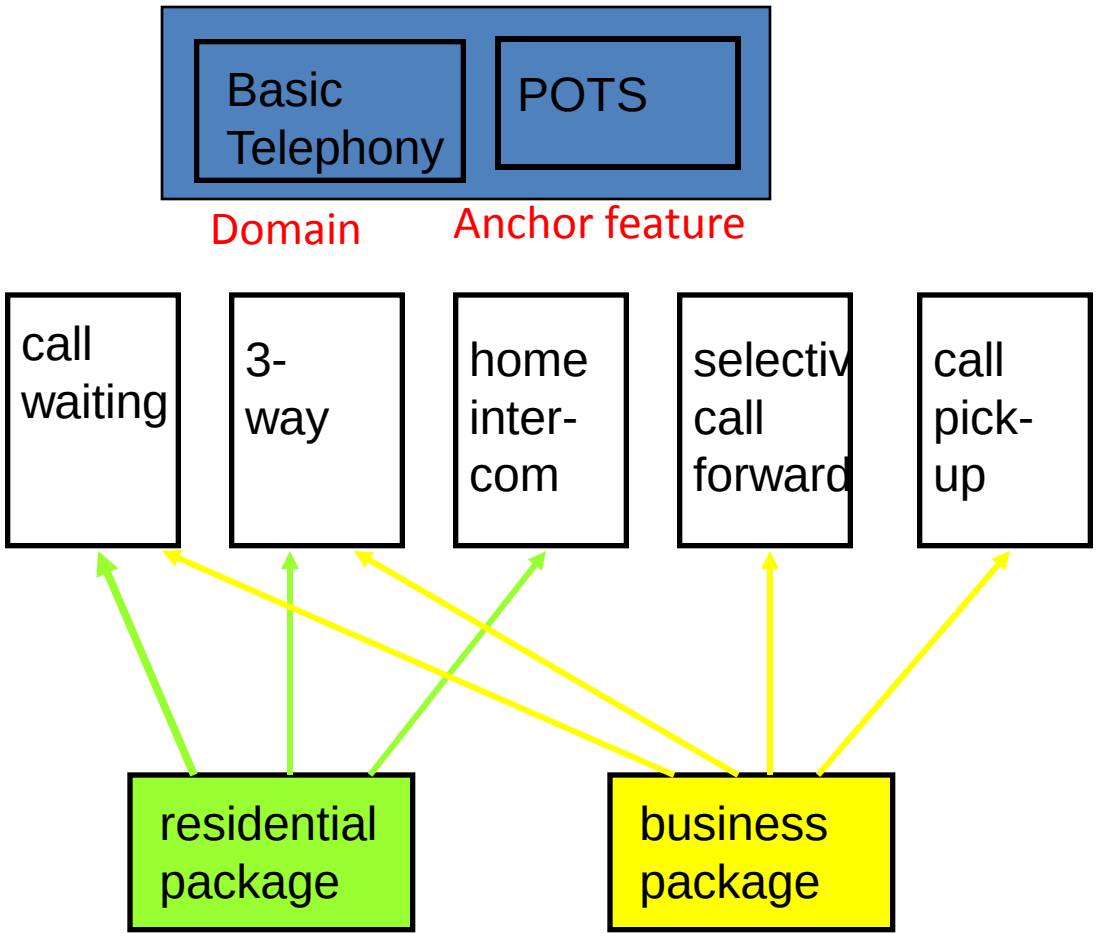


How hard is it to change code?



- When a function throws a new exception, programs that call the function may have to be changed
- Programs that transitively (indirectly) call the function may also have to be changed
- Existing programming languages offer very little help to programmers
- **Many popular programs crash often because exceptions are not handled**

Features and feature packages written in FLX are reusable



- Features are written based on a model instead of the code of other features
- Features and feature packages are integrated in a feature package
- One can integrate different combinations of features and feature packages into different feature packages

Challenges

- Automatic assertion based software verification
  - Grand Challenge (Hoare 2003)
  - Holy Grail of Computer Science (Gates 2005)
- Changing code is as hard
  - Assertion verification is to determine whether an assertion is true at some point in the code.
  - Finding out where to make changes is to determine where in the code an assertion becomes true

Getting to the Holy Grail

- We can automate assertion based verification of hardware circuits
  - Finite state machine
  - Boolean logic (Advances in SAT solvers in the late 90's)
- But with Software
  - State variables may not be bounded (e.g. integer)
    - FLX compiles an application into a FSM
  - Must reason on predicate logic whose variables have complex data structures (e.g. Is the linked list empty?)
    - A first order solver for assertions written in Java has been implemented

What are we working on?

- The FLX compiler.
- The FLX satisfiability solver for first order formulas
- A call center developed using
- A set of self healing features