Runtime GUI Adaptation in Dynamic Software Product Lines

Dean Kramer
deankramer@acm.org

University of West London/Middlesex University

FOSD 2014

Agenda

Background I

- Smart phones have been highly prolific.
- Large array of applications available
- Typically contain GPS, Internet, Compass, Light, Accelerometer sensors.
- Sensor data can gain a wide range of contextual information, that can be consumed by context aware applications.

Background II : Software Product Lines (SPL)

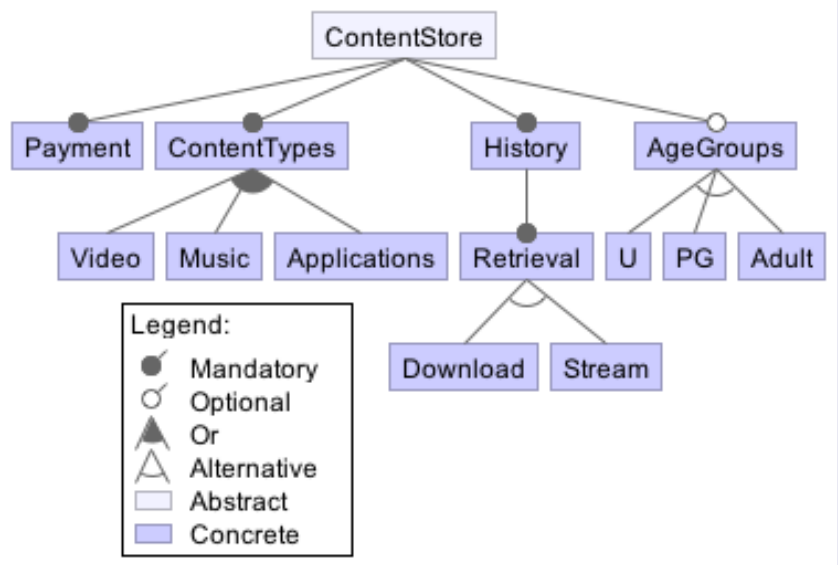
- Develop similar software from common assets
- Feature-Oriented Software Development as a method for modularising system features
- Dynamic Software Product lines (DSPL)
  - Runtime feature binding
  - Unified adaptation

Problem - GUI Variability

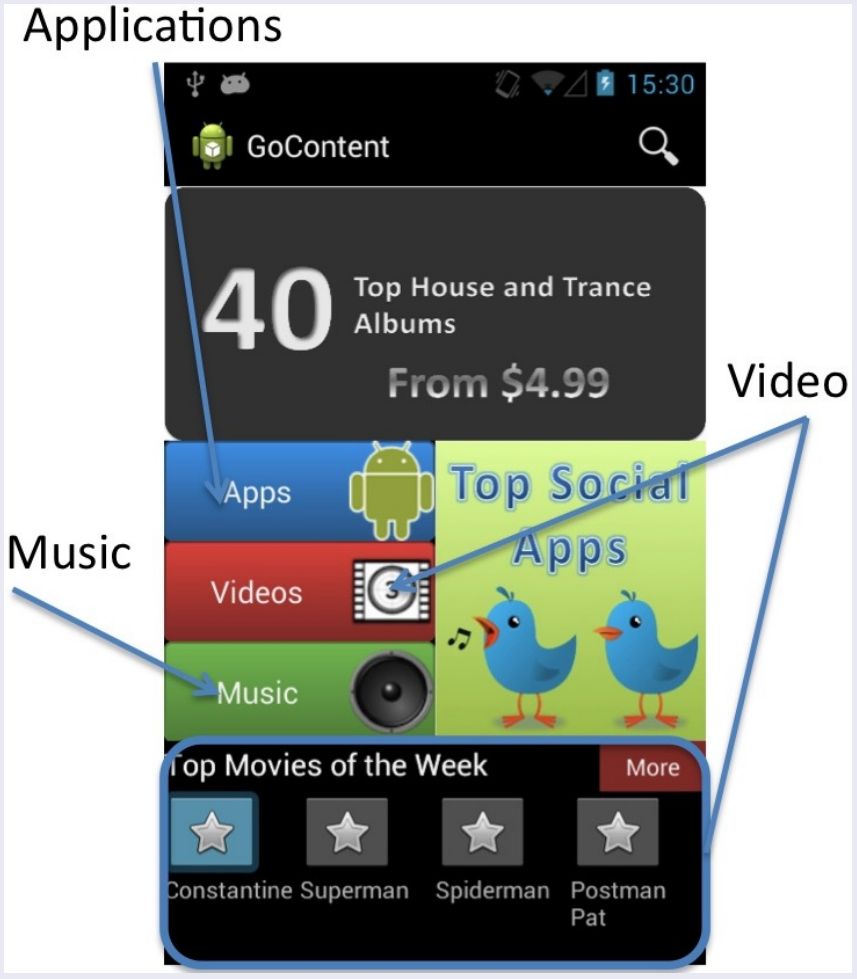
- Increasingly popular to develop GUI applications using more than general purpose languages
- GUIs can exhibit variability in SPLs
- Not all GUI variability can be statically realised:
  - Adaptive GUIs
  - Plastic UIs
- Logic adaptation already possible with DSPLs
  - Normally only single language solutions
  - No dedicated GUI support

Scenario - Content Store Application

Feature Model



Screen Variability



Document-Oriented GUIs

- DSLs for declaring GUI structure, and properties
- Also known as:
  - GUI Description Languages
  - GUI Markup Languages
- Different implementations:
  - Android (XMLBlock)
  - iOS & OSX (XNib/Nib)
  - Microsoft XAML
  - Mozilla XUL
  - QT QML

Android GUI Document Excerpt

```
<FrameLayout
  android:id="@+id/mainFrame"
  android:layout_width="match_parent"
  android:fitsSystemWindows="true"
  .....>
  ....
  <Button
    android:id="@+id/applications"
    android:layout_width="160dp"
    android:text="@string/apps"
    android:background="@drawable/apps"/>
  <LinearLayout
    android:id="@+id/adverts"
    ....>
    <TextView
      android:id="@+id/appAdsTitle"
      android:text="@string/PopularFreeAppss"/>
    ...
  </LinearLayout>
</FrameLayout>
```

Previous Work Recap: GPCE 2013 Paper

- Variability implemented in refinements.
- Use Dynamic Binding Units
- Generate GUI document variants at compile-time using Superimposition:
  - Based on document refinement combinations, not just feature combinations.
  - All combinations are checked for FM satisfiability

Previous Work Recap: GPCE 2013 Paper con.

- Code Generation and Transformation
  - Generate variant management code
  - Transform source code to call the variant management when a GUI document is used.
- Adaptation only handled when GUI document is needed (when it is created).
- No full adaptation once the GUI is visible to the user.

Runtime Adaptation Overview

- Full runtime adaptation now handled for:
  - GUI Document related
  - Source code related
- Adaptation can be handled either:
  - On inflation: When the GUI is first created (not program, but a particular screen).
  - When Active: After a GUI is already visible.
- Two class methods used with FOP to assist runtime adaptation:
  - Document Initialisation Methods
  - Other GUI Adaptations

Methods for Runtime Adapt.: Document Initialisation

- Used for all operations related to the GUI initialisation
- Can be also invoked in the GUI controller constructor
- Can be refined like other methods
  - Add initialisation operations when adding an additional widget

Video feature example for homepage

```
public void onCreate_homescreen(ViewGroup vg){
original();
Button btnVideo = (Button)vg.findViewById(R.id.video);
  btnVideo.setOnClickListener(new onClickListener() {
    public void onClick(View v) {
      gotoVideoStoreScreen();
    }
  });
}
```

Methods for Runtime Adapt.:Other GUI Adaptations

- GUI adaptation can be implemented in sourcecode including visual and nonvisual e.g. gestures
- Contained in onGUIConfiguration methods
- Can be refined also
- Is not automatically reversed!

Example operations to implement a List "swipe to remove"

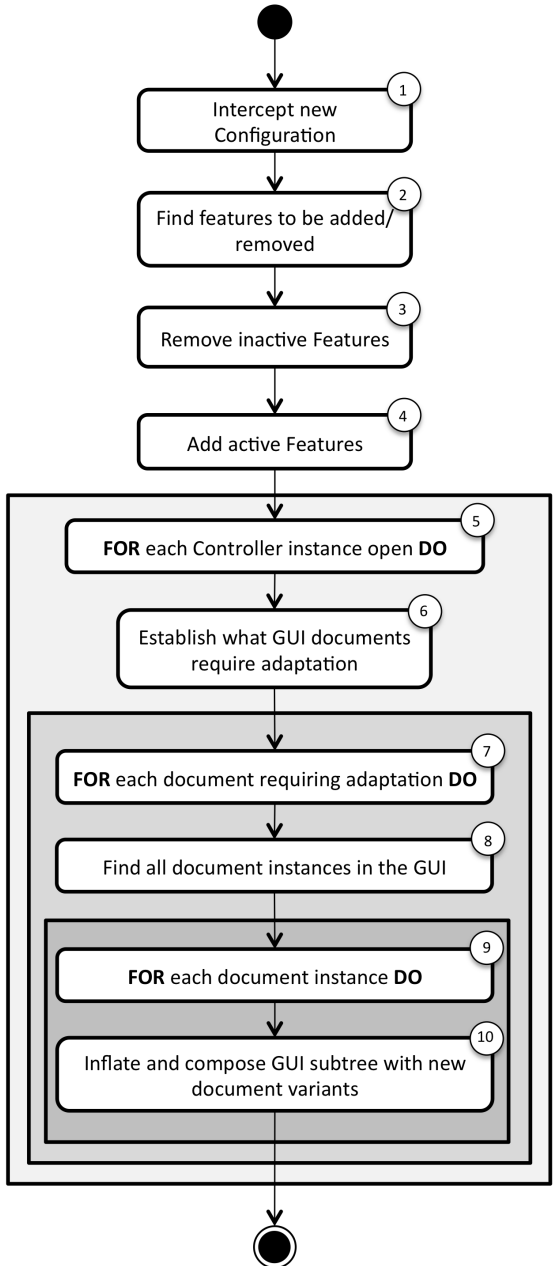
```
public void onGUIConfiguration() {
  SwipeDismissListViewTouchListener touchListener =
    new SwipeDismissListViewTouchListener(
      listView, new SwipeDismissListViewTouchListener.OnDismissCallback() {
          public void onDismiss(ListView listView, int[] reverseSortedPositions) {
            for (int position : reverseSortedPositions) {
              adapter.remove(adapter.getItem(position));
            }
            adapter.notifyDataSetChanged();
          }
        });
  listView.setOnTouchListener(touchListener);
  listView.setOnScrollListener(touchListener.makeScrollListener());
}
```

Generation & Transformations

- Three additional types of components generated:
  - Adaptation Manager - Orchestrates the adaptation.
  - State Transfer Component - For ensuring state is transfered between variants.
  - Feature classes (similar to FeatureC++) - Which hold data needed for the GUI adaptation.
- Further code transformations within the GUI controllers to handle the variant reloading, and tree composition.

Adaptation Process

- Runs over all currently active controllers (Android activities and fragments).
- Can handle adaptation of multiple documents in a reconfiguration.
- Adapts each instance of that GUI tree (important for lists!)



Implementation

- Developed to handle applications for the Android platform
- Built on top of FeatureIDE (No extension name yet)
- Static composition handled using FeatureHouse[1]
- FeatureDroid (DSPL Middleware)[2] used for handling context-acquisition and runtime configuration management

[1] https://github.com/deankramer/featurehouse

[2] http://deansserver.co.uk/gitweb/?p=AndroidDSPLMiddleware.git;a=summary

Scaling & Performance

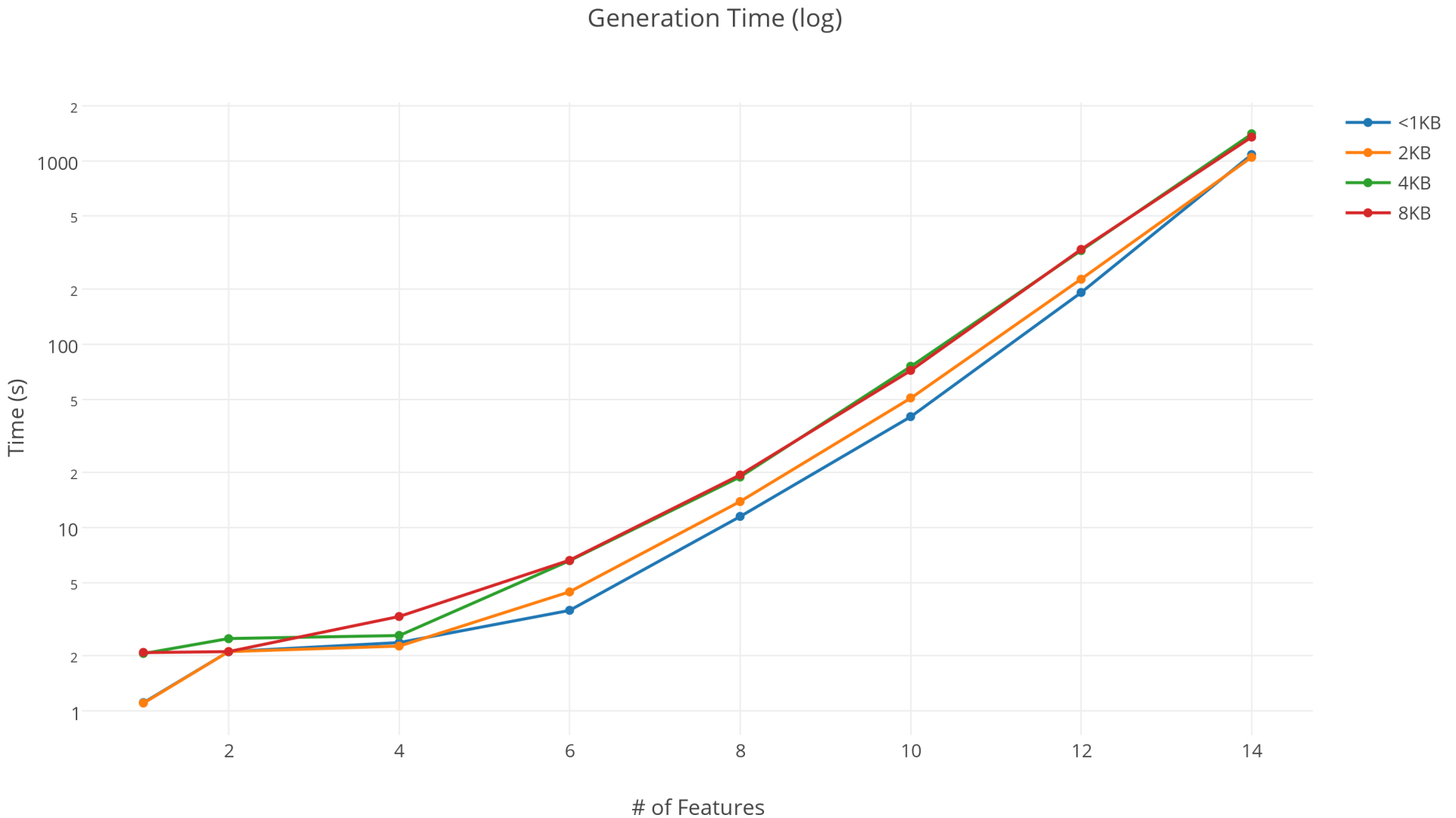
Examined how GUI document variability affects:

- Variant and support code generation time.
- Application size.
  - The size of the installation file (Android .apk file)
  - The installation size once installed on device
- Runtime adaptation time.
  - Base GUI document->Variant with all active features->Base GUI document.
  - Average time of 1000 adaptation cycles.

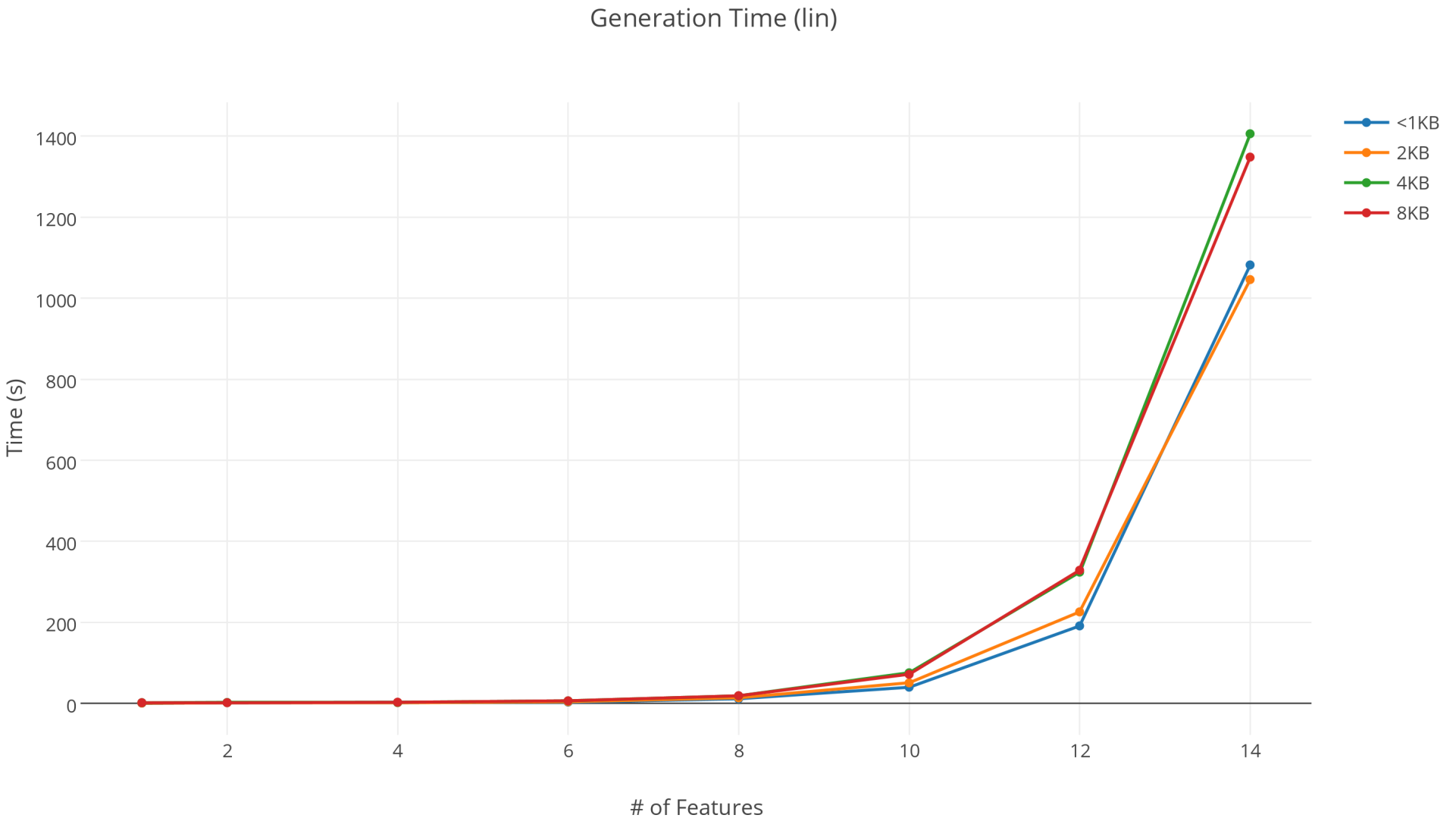
Scaling & Performance con.

- Tested different generated GUI document sizes including <1KB, 2KB, 4KB, 8KB
- Each refinement adds an additional button to the GUI document
- Refinements contained in optional features.
- Test machines:
  - Intel i5 laptop with 8GB of ram, Windows 8.1, standard HDD.
  - Nexus S phone running Android 4.2.1 - Single core, 3 1/12 years old.

Generation Time (Log)

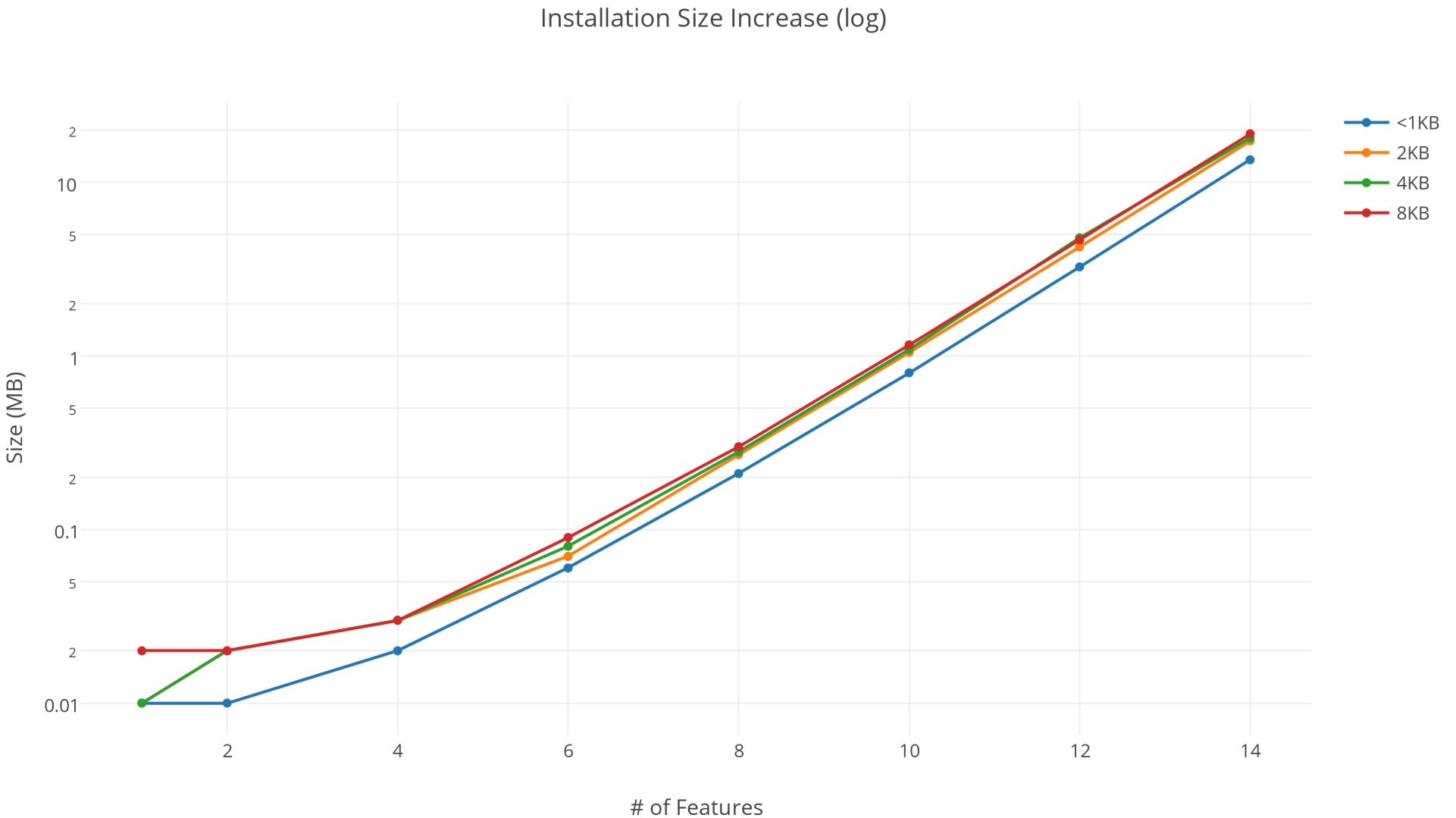


Generation Time (Lin)

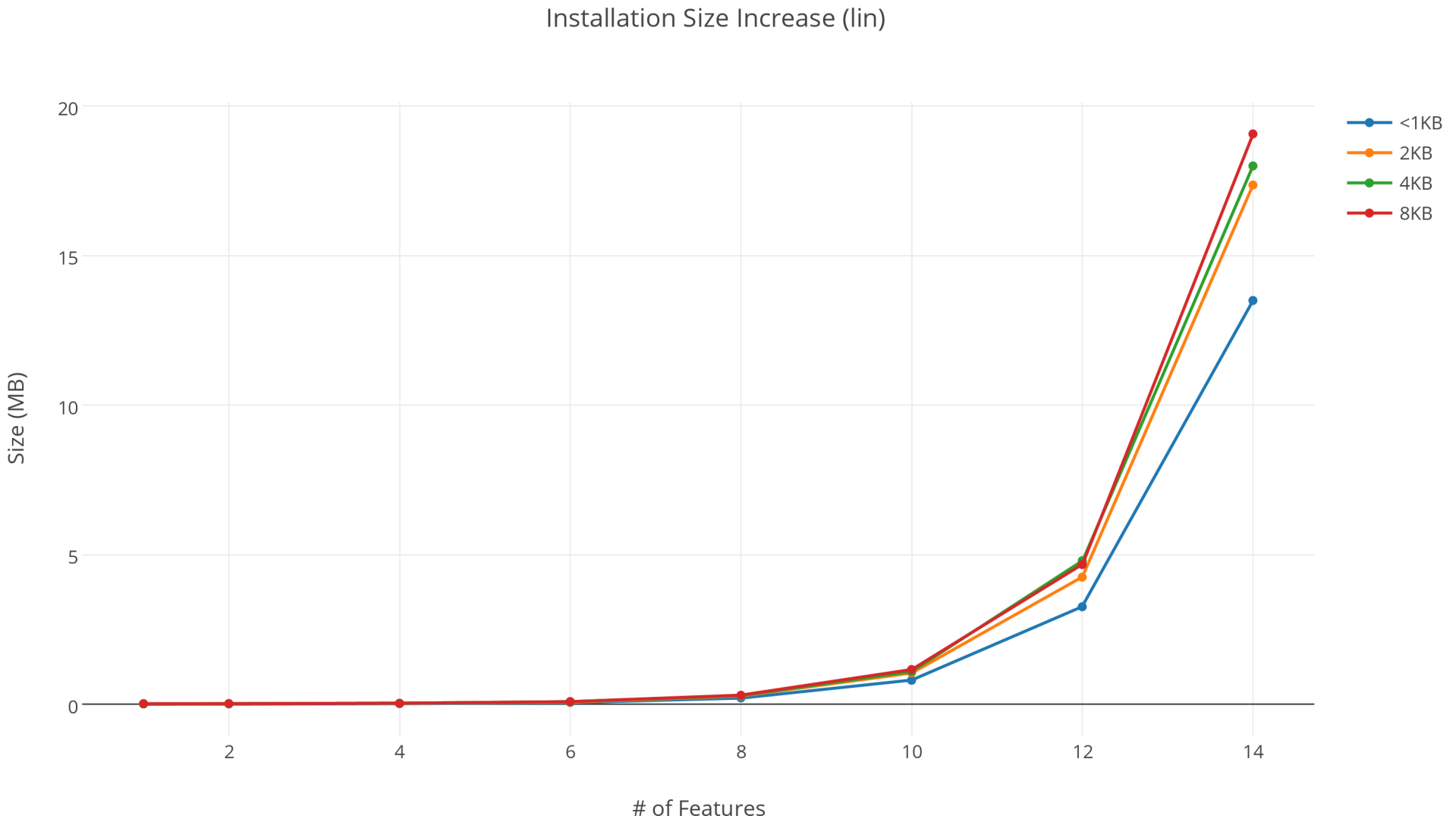


Source: generationtime

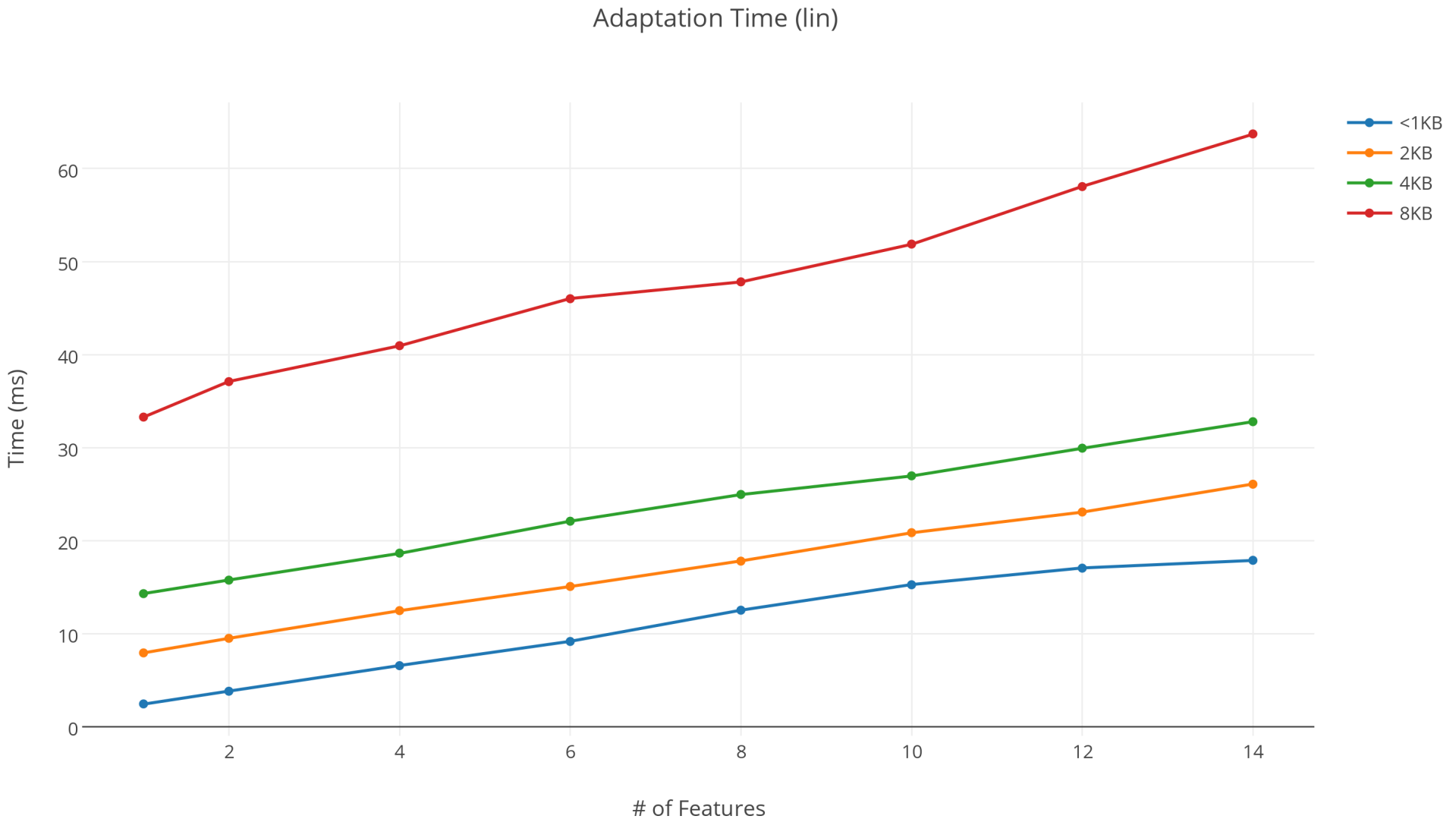
Application Installation Size Increase (Log)



Application Installation Size Increase (Lin)



Runtime Adaptation Time



Conclusions

- GUIs can exhibit both static and dynamic variability, which should be unified
- Our updated work allows for full runtime adaptation of the GUI
- Implementation shows promising feasibility for GUI documents
  - Feasible to 10-12 dynamic binding units
  - Still room for optimisation.

Q and A

Any questions?