Table 4. The Code of CICAppDoInitialization AsyncTask

```
class DoInitialization extends AsyncTask<Void, String, Void> {
  protected void onPreExecute() {
      super.onPreExecute(); }
  protected Void doInBackground(Void... arg0) {
      try {
          URL url = new URL("http://cic.hongik.ac.kr/_android/freetalk_select.php");
          HttpURLConnection conn = (HttpURLConnection) url.openConnection();
          . . .
          //Send to Server
          StringBuffer buffer = new StringBuffer();
          . . .
          //XmlPullParser Instance Object Creat for Recive Talk List
          . . .
          //XmlPullParsing Start
          . . .;
          String tag;
          FreetalkData data = new FreetalkData();
          while((parserEvent = xpp.next()) != XmlPullParser.END_DOCUMENT) {
              switch(parserEvent) {
                  case XmlPullParser.START_TAG:
                  .. case XmlPullParser.END_TAG:
                  . . . f(inTitle2) {
                      data.setText(xpp.getText());
                      . . .
                      if(data.getId().equals(user.getId())) {
                          publishProgress(null, data.getText(), null, "0");
                      } else {
                          publishProgress(data.getId(),data.getText(),data.getName(),"1");
                      }
                  }
                  break;
              }
          }
          conn.disconnect();
      }
      catch (IOException e) { e.printStackTrace();}
      catch (XmlPullParserException e) { e.printStackTrace(); }
      return null; }
}
```

V. CONCLUSION AND FUTURE RESEARCH

This study suggests building up automatic static analysis mechanism for mobile application SW testing, and methods to reduce defects of mobile application SW through visualizing activity call graph. A static analysis oriented Tool-Chain process is suggested, and it can work quality control and visualization enhanced by the previous static analysis of SW code.

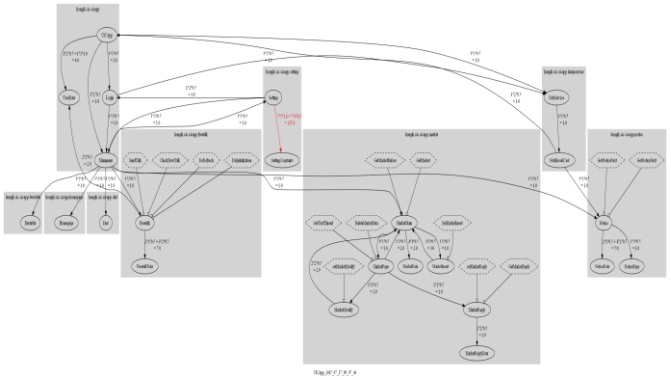

Figure 8: Visualization for asynchronous classes with dotted hexagon

Android codes are divided into internal function and UI. When the existing Tool-Chain process analyzes android mobile application SW, it is important to call and transfer data between activities as the basic unit of android UI that can't be analyzed and visualized. We suggest setting up the static analysis mechanism, and an activity call graph of android mobile SW application. This reduced coupling of mobile application SW, and removed unnecessary calls and data transmission by visualizing calls between activities. For future study, we are still keeping activity-related UI design, and reducing McCabe's cyclomatic complexity of coupling & cohesion.

ACKNOWLEDGMENT

This work was supported by 2014 Hongik University Research Fund and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2011601).

REFERENCES

[1] Park, B.K., Kwon, H.E., Yang, H.S., Moon, S.Y., Kim, Y.S. and Kim, R. Y.C., 2014, "A Study on Tool-Chain for statically analyzing Object Oriented Code," Proc. of Korea Computer Congress 2014, pp. 463-465.
[2] National IT Industry Promotion Agency (NIPA), http://www.nipa.kr
[3] Gao, J., Bai, X., Tsai, W. T. and Uehara, T., 2014, "Mobile Application Testing: A Tutorial," Computer, 47(2), pp. 46-55.
[4] Collberg, C., Kobourov, S., Nagra, J., Pitts, J. and Wampler, K., 2003, "A System for Graph-Based Visualization of the Evolution of Software," Proc. of The SoftVis '03 ACM symposium on Software visualization, pp. 77-86.
[5] Lanza, M., 2001, "The Evolution Matrix: Recovering Software Evolution using Software Visualization Techniques," Proc. of The IWPSE '01, pp. 37-42.
[6] Kang, G.H., Yi, K.S., Kim, D.H., Hwang, J.S., Kim, Y.S., Park, Y.B. and Kim, R. Y.C., 2014, "A Practical Study on Tool Chain for Code Static Analysis on Procedural Language," Proc. of Korea Computer Congress 2014, pp. 559-561.
[7] Kwon, H.E., Park, B.K., Yi, K.S., Park, Y.B., Kim, Y.S and Kim, R. Y.C., 2014, "Applying Reverse Engineering through extracting Models from Code Visualization," The 2014 Fall Conference of the KIPS, 21(2), pp. 650-653.
[8] Kang, G.H., Kim, R. Y.C., Yi, K.S., Kim, Y.S., Park, Y.B. and Son, H.S., 2015, "A Practical Study on Code Static Analysis through Open Source based Tool Chains," KIISE Transactions on Computing Practices, 21(2), pp. 148-153.
[9] SN group, Source Navigator User Guide [Online], Available: http://sourcenav.sourceforge.net/
[10] SQLite, About SQLite [Online], Available: https://www.sqlite.org