Advances in Social Science, Education and Humanities Research, volume 410

1st International Multidisciplinary Conference on Education, Technology, and Engineering (IMCETE 2019)

Software Testing as Quality Assurance on Web Application

1st Henderi
Faculty of Science and Technology
Raharja University
Jenderal Sudirman St. 40, Cikokol,
Tangerang, Indonesia, 15117
henderi@raharja.info

2nd Anrie Suryaningrat
Faculty of Science and Technology
Raharja University
Jenderal Sudirman St. 40, Cikokol,
Tangerang, Indonesia, 15117
anrie@raharja.info

***Abstract*—A web based application can be robust, scalable, easily maintained, and easy to use, but if it does not do what the customer needs then it is useless. The success of a new product or software application is at the mercy of how precisely it fulfills the requirements of its users. On by request software with custom criteria it should ensure that it meets the requirements of the end users. Otherwise the consequences will be likely to be failure, poor consumer experience, brand deterioration and major financial loss because of having to identify defects and fix them. This paper illustrates the benefits of code-level measurement efficiency to gain software quality assurance. After testing is done, we found the data were based function are 1.487 functions. There is only 52.25% functions really involved, 12.57% functions detected as duplicate, 41.22% functions must be rejected and 21.32% functions marked as deferred. These evidences aligment about findings for how software testing is organized. The developer have to a high degree of self-responsibility for quality assurance on code level.**

Keywords:* *application, code-level measurement, efficiency

I. Introduction

Software-supported environments can be critical, pervasive, persistent, mobile, distributed, real-time, context-aware, and adaptive. A growing need emerges for fast and rigorous approaches to develop and evolve these systems [15]. Software quality assurance is a process for guesstimating and documenting the quality of the software products during each phase of the software development lifecycle [17].

This research is based on problem that being happen on public service web-based application software XYZ on local government institution that shows problems in functionality. The problems found under black box testing. For illustration, the system have some group user roles, for each division and public access. Each group role have their own interface and modules, such as group Super Administrator have modules and interface to create access logged in for other users in other group roles.

Group Inspection and Administration have modules and interface for their inspections, controlling and administrative works respectively and user companies have modules and interface to input company data and their regular reports. What is mentioned as anomalies are something like there are no user access control between group roles, some modules form can not create input data or in other case have put wrong data into database table, modules that give wrong output or some missing important functionality like there is no edit or delete data. The complexity and the nature of systems and engineering processes in this industry, there is a strong need yet a slow shift toward innovation in software quality management [11]. Software testing-resource allocation plays a significant role in software project management [13].

Those conditions are not what users expected, since government institution users works become hampered, but also make public service cannot meet the required standard time and excellency and also affected to companies as user public for their business. The presence of design flaws in a software system has a negative impact on the quality of the software, as they indicate violations of design practices and principles, which make a software system harder to understand, maintain, and evolve. As mentioned in related work [10], exploratory testing plays an important role, the overall degree of automation is generally low due to resource and time limitions. In short words, software defects are tangible effects of poor software quality [5].

This research is not focused on user acceptance or user interface, or user experience, but further more about how something critical functions can be missed on development. This research also to find out if there are redundant components or may be some unused components in the software and how to give a recommendation for future works on this software development.

II. Method Overview

In this research, considered using combination between black-box and white-box testing on web-based application made with PHP framework based on Model View Controller in Figure 1. Black-box testing used to get information how modules in system behave such as URL page, while white-box testing as direct inspection on lines of code to find what cause the behavior. It was likes Evolutionary Multi-Objective Algorithms (EMOAs) which have been applied to derive products for the variabil- ity testing of Software Product Lines (SPLs).

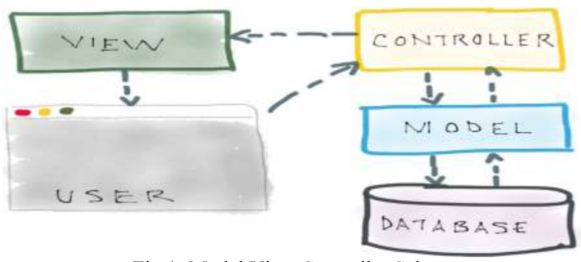


Fig 1. Model View Controller Scheme

Black-box testing used to get information how modules in system behave such as URL page, while white-box testing as direct inspection on lines of code to find what cause the behavior. It was likes Evolutionary Multi-Objective Algorithms (EMOAs) which have been applied to derive products for the variability testing of Software Product Lines (SPLs) [18].

Further more, main focus on testing is on White Box Testing, where the testing can be performed any time in the life cycle after the code is developed. In other words, White box testing is done at low level design and implementable code. White box testing can be applied at the unit, integration, and system levels of software testing process [1].

Implementation of white-box testing can be done, because scope of work author as maintenance programmer for this application. So that, full access to code is available and thorough inspection can be made. Even more, since white box testing can be wide spread and time consuming. In order to scope of testing keep maintained and efficient the test needs to be designed in a way that the number of test cases is sufficient and appropriate in quantity[2]. In this case, as the object are functions in Models and Controllers directories.

The research begin with ask the questions. Is it possible to remove unused lines of code which might be causing introduction of bug? How many functions that have role in the system? How about the logic flow written ? May there are using no-reuse model in development, so then causing more than just inefficiency in coding but also misplace data input into database? In order answer those questions, from data URL page there are information what class and methods in Controllers and Models involved. For example, from URL http://domain_name.tld/index.php/member/C_monitor there is information that page constructed from file and class C_monitor in Controller/member directory and using method index(). More over, from file C_monitor gathered information that the class component using file model M_monitor.

At a glance, nothing wrong with the code above. However after more inspection on code lines in Figure 1 above, found that bug defect on lines 7 - 9, where validation only checking if username is not null or empty, but there is no group role or level checking. This condition may impact on other methods inside class. In this way, user from group role company (public user) can access module that made specifically for group role Administrator or Inspection as government user.

Moreover, the URL synthesis is not use routing or masking. As security aspect, this bug defect may give freedom for attacker to make the system so messy and may collapse. This one of critical bug defect that being happened should be fixed as high priority. In other place, code in lines 10-12, show inefficiency since can be load via application autoload. On line 21-25 can be replace with using global configuration from application config or custom config file. On line 31-32, and 34 there are unused variable $data, and on line 31 and 34, redundant files for header and footer respectively.

Offered solution for this part is put validation on constructor for group user role and, removed unused codes, as seen on Figure 2. In Figure 2, the offered solution are: (a) do validation for user session on login, show on line 8-12, (2) remove libraries load (since have loaded from application autoload file) - line 15, (3) add load custom config on line 21 as described on line 20, (4) remove unused codes $data[‘username’], $data[‘sid’],$data[‘nama_user’], (5) remove unused variable $data from load view, and (6) reuse files header and footer. In file M_monitor, we can get information what functions inside, as displayed on Figure M_monitor.

There is something interesting on file M_user.php that show some functions similarity with file M_monitor.php. As seen on code attached above, there are repetition methods, as seen on lines 8-47 on file M_monitor and lines 12-47 on file M_user.php. This findings give curious if there are more duplicate functions and or may be functions that suspect as rejected in the system.

In other place, for example why company data show on module user Inspection is different from data list on administrator. Logically, any company profile data for every module refer to master data made by group user Administrator. What make this happen, begin on input data form, where any data input, including company data is get into table database. This is not only causing redundant data on database, other thing may happen is if there is change on master data for company, then make different data. Other thing looks interesting.

After checking on file C_monitor, redundant data on database was triggered by variables data on line 3-11, 16-18, 22-29, 35-37. Another wrong procedure also triggered on line 40-50, where data input also inserted into master data company and this will cause invalid data since master data should only come from user with group role Administrative. In other case, some bug defect for example also shown on on lines 6 - 13 in Figure 6. The SQL query takes all data from table without checking if company data was exists in table business entity. This may cause redundancy and the worst scenario may give not valid data if master data company was changed.

The situation above can be avoided filtering data input, then only specific data get into database, while for related data can be use join with master data company table, as shown on proposed revised code shown on Figure 7. In Figure 7, data input was filtered as shown on line 10-16, any redundant data and mis-logic procedure as seen on Fiure 5 was removed. For display data, revised function on M_pengawasan may like on Figure 8. In order to give valid data output to display, SQL Query was done using join with master data company as seen on lines 8-22 (Figure 8). Bug defects as shown on figures above, was example how testing was done.

Next, the result for this case study research using test metric to give the report. If metrics are not followed, then the work completed by the test analyst will be subjective i.e. the test report will not have the proper information to know the status of the work/project. If metrics are involved in the project, then the exact status of the work with proper numbers/data can be published [8].

Further more testing process will checking the code manually, counting how many functions involved, canditate to rejected, duplicate, and deferred. Data result will be compare into reputation table where scale is from 0% to 100% in 4 level and make some recommendation.

TABLE 1. REPUTATION LEVEL

Level	**Score (%)**
Excellent	75 - 100
Good	50 - 74.99
Poor	25 - 49.99
Very poor	0 - 24.99

In ISTQB, test analysis is described as during test analysis, the test basis documentation is analyzed in order to determine what to test, i.e., to identify the test conditions. A test condition is defined as an item or event that could be verified by one or more test cases (e.g., a function, transaction, quality characteristic or structural element). Also, a test case that is to be the output of the test design is described as a test case is developed to cover a certain test objectives or test conditions [9].

III. Results and Analysis

After testing process is stopped, found the data are based function (BS) are 1.487 functions, rejected functions (RJ) are 613 functions, duplicate functions (DP) are 187 functions, real functions involved in system (RD) are 710 functions, and deferred functions (DR) are 317 functions. While the data is plotted into reputation table, we can have information as show on Table 2.

TABLE 2. COMPARISON RATIO PERCENTAGE

Num.	**Comparison**	**Result (%)**	**Level**
1	RD vs BS	47.75	Poor
2	DR vs BS	21.32	Very poor
3	RJ vs BS	41.22	Poor
4	DP vs BS	12.57	Very poor
	Average	**30,72%**	**Poor**

Based on Table 2, there is only 52.25% functions really involved. However, there were 12.57% functions detected as duplicate, 41.22% functions must be rejected, and 21.32% functions marked as deferred. These evidences aligment about findings for how software testing is organized [x1]. The developer have to a high degree of self-responsibility for quality assurance on code level.

A set of 33 (out of 81) practices were ranked as mandatory by most of participants, which represents 40% of the TMMi's full set of practices; on the downside, a testing process that relies on this subset of TMMi practices does not fully fulfil level 2 (managed) of the maturity model [12]. Performance bottlenecks were found automatically and confirmed by experienced testers and developers [14].

IV. Discussion

Since the scope of research was doing retest the software after released. If there are any bug defects happen along usage by customer will be recommendation for next step on maintenance phase or may become re-engineering if needed. Programs are coded by calling a set of functions. Each function is a smallest unit that does a specific functionality. It specifies much functionality of the design has been exercised by the verification environment.

Tests are written to exercise each of the different functions in the code, thus it is proven that if each test is completed properly, then entire set of functionality is verified. This test aligment with the ISO//IEC25010 standard quality attributes of a software product a likes in Carrozza [16] about internal quality attribute. However, this coverage has 2 limitations. That are, there is not a defined list of 100% functionality of the design is and therefore, there may be a missing functionality in the list, and there is no real way to check that the coverage model is correct, manual check is the only way [3].

The major motive of testing can be assurance of quality, estimation of reliability, validation and verification. Software testing is an elemental constituent of software quality assurance and represents a review of specification, design and coding. The major target of software testing is to affirm the quality of software system by systematically testing the software in carefully controlled circumstances, another aim is to recognize the completeness and correctness of the software, and ultimately it reveals undiscovered errors. Testing is basically a task of locating Errors [4]. However, systems adopting a specific architecture (e.g., the Model-View-Controller pattern) can be affected by other types of poor practices that only manifest themselves in the chosen architecture [6].

V. Conclusion

Bug life cycle is a complete life cycle of a bug. Code bugs found because of wrong code, missing code, the particular feature is not available in the application, and extra coding. Based on 1.487 functions in a web application, thie reseach found 52.25% functions really involved, 12.57% functions detected as duplicate, 41.22% functions must be rejected and 21.32% functions marked as deferred. These evidences means the developer have to a high degree of self-responsibility for quality assurance on code level. Further works to do is recommendation for more performance testing, for scale-ability based on code, logic flow and consistency between components and reuse components.

VI. References

[1] Manish Kumar 1 Santosh Kumar Singh, A Comparative Study of Black Box Testing and White Box Testing Techniques (2015).

[2] Tsuyoshi Yumoto, Toru Matsuodani, Kazuhiko Tsuda, A Test Analysis Method for Black Box Testing Using AUT and Fault Knowledge (2013)

[3] Nidhi Gupta, Different Approaches to White Box Testing to Find Bug (2014)

[4] Neetu Dhingra, Mayank, Contingent study of Black Box and White Box Testing Techniques (2014)

[5] Marco D'Ambros, Alberto Bacchelli, Michele Lanza, On the Impact of Design Flaws on Software Defects(2010)

[6] Maurício Aniche, Gabriele Bavota, Christoph Treude, Marco Aurélio Gerosa, Arie van Deursen, Code smells for Model-View-Controller architectures (2017)

[7] Arti Rana,Arvind Singh Rawat,Anchit Bijalwan, Process of finding defects in software testing (2017)

[8] T John Vijay , Dr. M. Gopi Chand , Dr. Harika Done, Software Quality Metrics in Quality Assurance to Study the Impact of External Factors Related to Time (2017)

[9] ISTQB FLWG. "Foundation Level Syllabus Version 2011": International Software Testing Qualifications Board ; 2011

[10] Felderer M., and Ramler R., Risk Orietntation in Software of Small and Medium Enterprises: an Exploratory and Comparative Studi, *Software Qual J (2016), Springer.* pp. 519-548.

[11] Carrozza G., Pietrantuono R., and Russo S., A Software Quality Frameworkd For Large-scale Mission-Critical System Engineering, *Information and Software Technology Journal* **(102)**, Elsevieer, 2018, pp. 100-116.

[12] Camargo G. K., Ferrari C. F., and Fabbri CPF S., Characterising The State of The Prcatice in Software Testing Through a TMMi-based Process, *Journal of Software Engineering Research and Development* (2015) 3:7., Springer.

[13] Xiao X., Dohi T, and Okamura H., Optimal Software Testing Resource Allocation With Operational Profile: Computational Aspects, *Journal Life Cycle Reliability and Safety Engineering*, 2018, 7, Elsevier, pp: 269-283.

[14] Luo Q, Nair A., and Grechanik M., FOREPOST: Finding Performance Problems Automatically With Feedback-directed Learning Software Testing, *Empir Software Eng. Journal*, 2017, 22:6–56, Springer Science -Business Media New York.

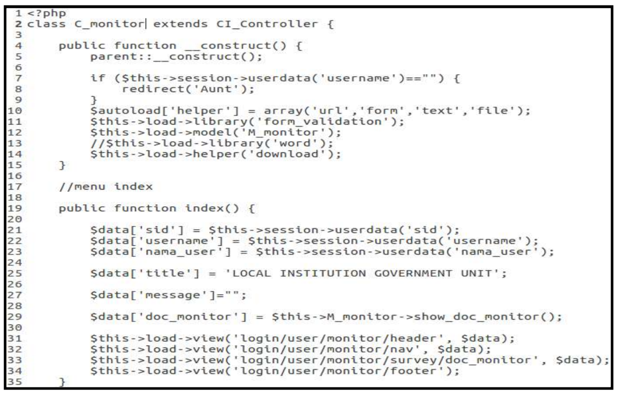
[15] Endo T.A., Bertolino A., Maldonado C.J., and Delamaro E.M., Guest Editorial Foreword for The Special Issue on Automated Software Testing: Trends and Evidence, *Journal of Software Engineering Research and Development*, 2018, 6:2, Springer.

[16] Carrozza G., Pietrantuono R., and Russo S., A Software Quality Framework For Large-Scale Mission-Critical Systems Engineering,*Journal Information and Software Technology* **102** (2018)*, pp. 100–116,* Elsevier.

[17] Vijay J., Chand G.M., and Done H., Software Quality Metrics in Quality Assurance to Study the Impact of External Factors Related to Time, *International Journal of Advanced Research in Computer Science and Software Engineering,* **Vol. 7 Issue 1**, 2017, pp. 221-224.

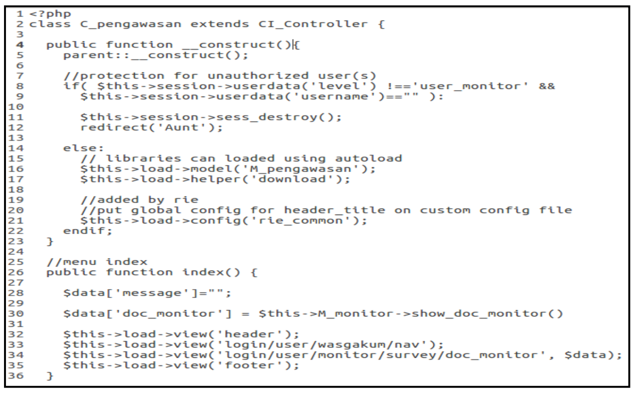
[18] Filho J.L.H., Ferreira N.T., and Vergilio R.V., Preference based multi-objective algorithms applied to the variability testing of software product lines, *The Journal of Systems and Software* **(151)**, 2019, pp. 194–209, Elsevier.

APPENDICES



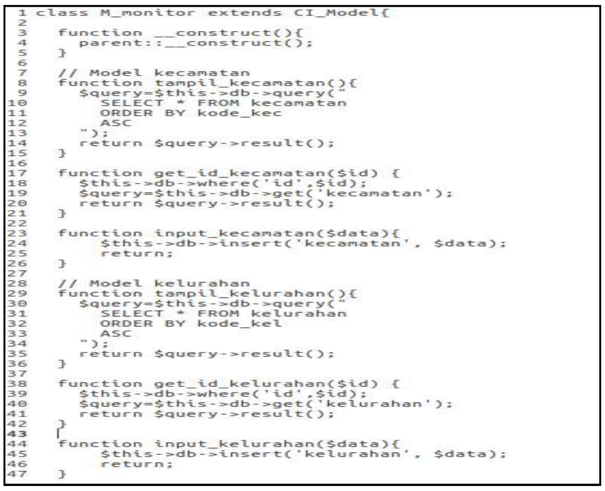
```
<?php
class C_monitor extends CI_Controller {

    public function __construct() {
        parent::__construct();

        if ($this->session->userdata('username')=="") {
            redirect('Aunt');
        }
        $autoload['helper'] = array('url','form','text','file');
        $this->load->library('form_validation');
        $this->load->model('M_monitor');
        //$this->load->library('word');
        $this->load->helper('download');
    }

    //menu index

    public function index() {

        $data['sid'] = $this->session->userdata('sid');
        $data['username'] = $this->session->userdata('username');
        $data['nama_user'] = $this->session->userdata('nama_user');

        $data['title'] = 'LOCAL INSTITUTION GOVERNMENT UNIT';

        $data['message']="";

        $data['doc_monitor'] = $this->M_monitor->show_doc_monitor();

        $this->load->view('login/user/monitor/header', $data);
        $this->load->view('login/user/monitor/nav', $data);
        $this->load->view('login/user/monitor/survey/doc_monitor', $data);
        $this->load->view('login/user/monitor/footer');
    }
```

Fig 1. Controller Class and Method



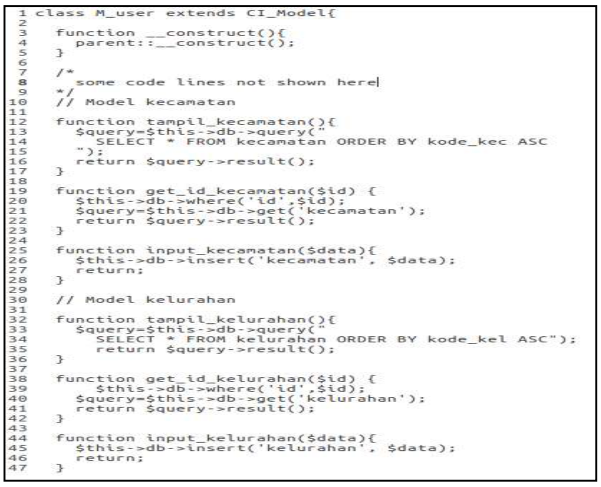
```
<?php
class C_pengawasan extends CI_Controller {

  public function __construct(){
    parent::__construct();

    //protection for unauthorized user(s)
    if( $this->session->userdata('level') !=='user_monitor' &&
      $this->session->userdata('username')=="" ):

      $this->session->sess_destroy();
      redirect('Aunt');

    else:
      // libraries can loaded using autoload
      $this->load->model('M_pengawasan');
      $this->load->helper('download');

      //added by rie
      //put global config for header_title on custom config file
      $this->load->config('rie_common');
    endif;
  }

  //menu index
  public function index() {

    $data['message']="";

    $data['doc_monitor'] = $this->M_monitor->show_doc_monitor()

    $this->load->view('header');
    $this->load->view('login/user/wasgakum/nav');
    $this->load->view('login/user/monitor/survey/doc_monitor', $data);
    $this->load->view('footer');
  }
```

Fig 2. C-monitoring.php - proposed revision



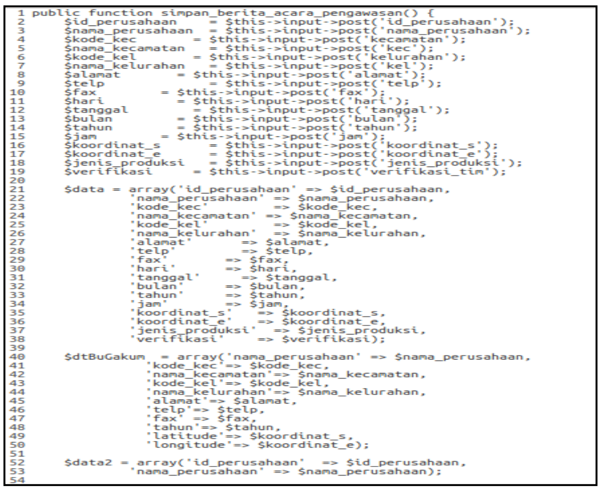
```
class M_monitor extends CI_Model{

  function __construct(){
    parent::__construct();
  }

  // Model kecamatan
  function tampil_kecamatan(){
    $query=$this->db->query("
      SELECT * FROM kecamatan
      ORDER BY kode_kec
      ASC
    ");
    return $query->result();
  }

  function get_id_kecamatan($id) {
    $this->db->where('id',$id);
    $query=$this->db->get('kecamatan');
    return $query->result();
  }

  function input_kecamatan($data){
      $this->db->insert('kecamatan', $data);
      return;
  }

  // Model kelurahan
  function tampil_kelurahan(){
    $query=$this->db->query("
      SELECT * FROM kelurahan
      ORDER BY kode_kel
      ASC
    ");
    return $query->result();
  }

  function get_id_kelurahan($id) {
    $this->db->where('id',$id);
    $query=$this->db->get('kelurahan');
    return $query->result();
  }

  function input_kelurahan($data){
      $this->db->insert('kelurahan', $data);
      return;
  }
```

Fig 3. File M_monitor - part 1



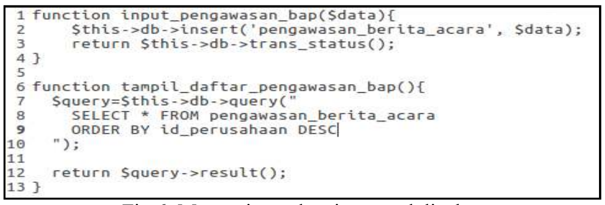
```
class M_user extends CI_Model{

  function __construct(){
    parent::__construct();
  }

  /*
    some code lines not shown here
  */
  // Model kecamatan

  function tampil_kecamatan(){
    $query=$this->db->query("
      SELECT * FROM kecamatan ORDER BY kode_kec ASC
    ");
    return $query->result();
  }

  function get_id_kecamatan($id) {
    $this->db->where('id',$id);
    $query=$this->db->get('kecamatan');
    return $query->result();
  }

  function input_kecamatan($data){
    $this->db->insert('kecamatan', $data);
    return;
  }

  // Model kelurahan

  function tampil_kelurahan(){
    $query=$this->db->query("
      SELECT * FROM kelurahan ORDER BY kode_kel ASC");
      return $query->result();
  }

  function get_id_kelurahan($id) {
      $this->db->where('id',$id);
    $query=$this->db->get('kelurahan');
    return $query->result();
  }

  function input_kelurahan($data){
    $this->db->insert('kelurahan', $data);
    return;
  }
```

Fig 4. M_user.php



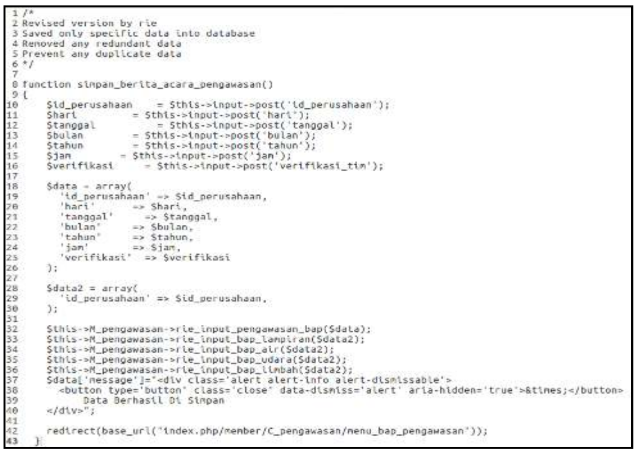
```php
public function simpan_berita_acara_pengawasan() {
    $id_perusahaan      = $this->input->post('id_perusahaan');
    $nama_perusahaan    = $this->input->post('nama_perusahaan');
    $kode_kec         = $this->input->post('kecamatan');
    $nama_kecamatan     = $this->input->post('kec');
    $kode_kel         = $this->input->post('kelurahan');
    $nama_kelurahan     = $this->input->post('kel');
    $alamat         = $this->input->post('alamat');
    $telp               = $this->input->post('telp');
    $fax          = $this->input->post('fax');
    $hari           = $this->input->post('hari');
    $tanggal          = $this->input->post('tanggal');
    $bulan          = $this->input->post('bulan');
    $tahun          = $this->input->post('tahun');
    $jam          = $this->input->post('jam');
    $koordinat_s        = $this->input->post('koordinat_s');
    $koordinat_e        = $this->input->post('koordinat_e');
    $jenis_produksi     = $this->input->post('jenis_produksi');
    $verifikasi       = $this->input->post('verifikasi_tim');

    $data = array('id_perusahaan' => $id_perusahaan,
            'nama_perusahaan' => $nama_perusahaan,
            'kode_kec'          => $kode_kec,
            'nama_kecamatan' => $nama_kecamatan,
            'kode_kel'          => $kode_kel,
            'nama_kelurahan'  => $nama_kelurahan,
            'alamat'        => $alamat,
            'telp'          => $telp,
            'fax'         => $fax,
            'hari'        => $hari,
            'tanggal'       => $tanggal,
            'bulan'       => $bulan,
            'tahun'       => $tahun,
            'jam'         => $jam,
            'koordinat_s'    => $koordinat_s,
            'koordinat_e'    => $koordinat_e,
            'jenis_produksi'  => $jenis_produksi,
            'verifikasi'     => $verifikasi);

    $dtBuGakum   = array('nama_perusahaan' => $nama_perusahaan,
              'kode_kec'=> $kode_kec,
              'nama_kecamatan'=> $nama_kecamatan,
              'kode_kel'=> $kode_kel,
              'nama_kelurahan'=> $nama_kelurahan,
              'alamat'=> $alamat,
              'telp'=> $telp,
              'fax' => $fax,
              'tahun'=> $tahun,
              'latitude'=> $koordinat_s,
              'longitude'=> $koordinat_e);

    $data2 = array('id_perusahaan'   => $id_perusahaan,
            'nama_perusahaan' => $nama_perusahaan);

```

Fig 5. C_monitor - function save data



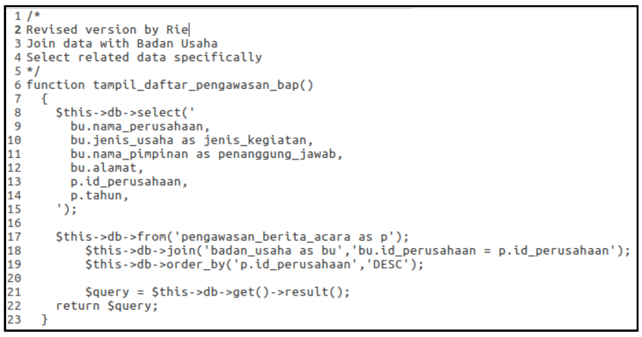
```php
function input_pengawasan_bap($data){
    $this->db->insert('pengawasan_berita_acara', $data);
    return $this->db->trans_status();
}

function tampil_daftar_pengawasan_bap(){
  $query=$this->db->query("
    SELECT * FROM pengawasan_berita_acara
    ORDER BY id_perusahaan DESC
  ");

  return $query->result();
}
```

Fig 6. M_monitor - data input and display



```php
/*
Revised version by rie
Saved only specific data into database
Removed any redundant data
Prevent any duplicate data
*/

function simpan_berita_acara_pengawasan()
{
    $id_perusahaan     = $this->input->post('id_perusahaan');
    $hari            = $this->input->post('hari');
    $tanggal             = $this->input->post('tanggal');
    $bulan           = $this->input->post('bulan');
    $tahun           = $this->input->post('tahun');
    $jam           = $this->input->post('jam');
    $verifikasi        = $this->input->post('verifikasi_tim');

    $data = array(
      'id_perusahaan' => $id_perusahaan,
      'hari'       => $hari,
      'tanggal'      => $tanggal,
      'bulan'      => $bulan,
      'tahun'      => $tahun,
      'jam'        => $jam,
      'verifikasi'  => $verifikasi
    );

    $data2 = array(
      'id_perusahaan' => $id_perusahaan,
    );

    $this->M_pengawasan->rie_input_pengawasan_bap($data);
    $this->M_pengawasan->rie_input_bap_lampiran($data2);
    $this->M_pengawasan->rie_input_bap_air($data2);
    $this->M_pengawasan->rie_input_bap_udara($data2);
    $this->M_pengawasan->rie_input_bap_limbah($data2);
    $data['message']="<div class='alert alert-info alert-dismissable'>
      <button type='button' class='close' data-dismiss='alert' aria-hidden='true'>&times;</button>
          Data Berhasil Di Simpan
    </div>";

    redirect(base_url('index.php/member/C_pengawasan/menu_bap_pengawasan'));
  }
```

Fig 7. C_monitor - proposed input data



```php
/*
Revised version by Rie
Join data with Badan Usaha
Select related data specifically
*/
function tampil_daftar_pengawasan_bap()
  {
    $this->db->select('
      bu.nama_perusahaan,
      bu.jenis_usaha as jenis_kegiatan,
      bu.nama_pimpinan as penanggung_jawab,
      bu.alamat,
      p.id_perusahaan,
      p.tahun,
    ');

    $this->db->from('pengawasan_berita_acara as p');
        $this->db->join('badan_usaha as bu','bu.id_perusahaan = p.id_perusahaan');
        $this->db->order_by('p.id_perusahaan','DESC');

        $query = $this->db->get()->result();
    return $query;
  }
```

Fig 8. M_monitor - proposed display dat