Thesis for the Doctor of Philosophy

Study on Your PhD Dissertation Title

Gildong Hong

Graduate School of Hanyang University

February 2022

Thesis for the Doctor of Philosophy

Study on Your PhD Dissertation Title

Thesis Supervisor: Jackson Smith

A Thesis submitted to the graduate school of
Hanyang University in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Gildong Hong

February 2022

Department of Computer Science & Engineering
Graduate School of Hanyang University

This thesis, written by Gildong Hong,
has been approved as a thesis for the degree of
Doctor of Philosophy.

February 2022

Committee Chairman: Jack Davis (Signature)

Committee Member: Ryan Tomas (Signature)

Committee Member: Nicholas Miller (Signature)

Committee Member: James Taylor (Signature)

Committee Member: Jackson Smith (Signature)

Graduate School of Hanyang University

Contents

List of Figures

List of Tables

Abstract

Study on Your PhD Dissertation Title

Gildong Hong
Department of Computer Science & Engineering
The Graduate School
Hanyang University

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris sed odio a tortor luctus placerat. Sed id eleifend ex, quis ornare orci. Aliquam at nisi lectus. Duis feugiat tincidunt pulvinar. Mauris nec tempor erat. Nullam feugiat sem et turpis tempus feugiat. Suspendisse et ante lectus. Mauris vitae nisi ut tellus gravida sodales. Ut gravida dignissim rhoncus. Ut porttitor libero sit amet lectus fermentum, ac laoreet felis aliquam. Curabitur convallis interdum enim et placerat.

Pellentesque sit amet ex dapibus, malesuada magna vitae, vulputate quam. Vestibulum et porta nulla. Donec massa nibh, tincidunt id turpis consectetur, pellentesque bibendum massa. Sed tincidunt id ligula sit amet faucibus. Donec vel tincidunt nunc. Sed vel finibus enim. Suspendisse vulputate interdum augue, et pulvinar sem molestie eget. Nulla facilisi. Aliquam sagittis volutpat tortor, vitae tempus dui commodo a. Fusce a diam in ligula rutrum malesuada id vel nibh. Proin metus magna, fermentum eu quam a, porta rhoncus enim. Donec a dui sit amet est pulvinar consectetur et ut justo.

Curabitur aliquet lacus at massa ultricies, et maximus arcu posuere. Cras dignissim velit enim, semper tincidunt mauris ornare id. Aliquam porttitor

nec dolor nec lacinia. Integer auctor lorem a nulla tristique viverra. Etiam a porta massa. Etiam vel ante eget nunc egestas ultrices quis quis enim. Nunc ex mauris, commodo nec nunc sed, pretium congue leo. Morbi vel blandit purus. Suspendisse potenti.

Pellentesque accumsan ultricies nunc id finibus. Sed maximus mauris ante, sit amet vehicula odio auctor at. Ut quis auctor tortor. Nulla hendrerit diam ac sapien faucibus bibendum. Morbi augue augue, semper non facilisis ac, ornare ut nibh. Vestibulum ut lacus fringilla, pretium nibh sed, cursus lorem. Curabitur malesuada orci a augue placerat ornare. Cras at dui et erat mollis cursus. Vestibulum sagittis scelerisque metus, vitae commodo arcu lacinia vel. Nullam pretium faucibus lobortis. Fusce eget blandit nisi, quis placerat justo. Maecenas condimentum quis erat at mollis. Curabitur nulla risus, varius vitae nibh vel, dignissim ullamcorper nunc. Vestibulum posuere pretium mauris vel tristique.

Chapter 1.Introduction

1.1 aaa

A Software Product Line (SPL) consists of the family of products that share the common set of features to accomplish the specific market segments. The main objective of SPL is to develop the family of software [?, ?, ?]. The family of software from SPL provides the auspicious way for the development of the large variety of software systems by the reusability of common and variable features from the core assets. Core asset of SPL consists of all required features that indicate the specification and scope of the family of software [?, ?, ?]. SPL is providing the best improvement in software industry due to the fast development of family of products by the reusability of features from core assets [?, ?].

Industry uses SPL such as the automobile, mobile phone, IoT applications etc. for the development of the family of products to achieve market segments. Industry claims that SPL provides the auspicious way for better, faster and cheaper development of the large variety of software systems. SPL organizations have reported enhancement in the order-of-magnitude related to quality, cost and time to market [?, ?]. All products that aim to develop from SPL are differed from each other according to the perspective of individual end-users or market requirements. To acquire the high quality, less development cost and time to market, SPL is based on the reusability of software components [?, ?].

Reusable components of SPL are common and variable features. These features are used to develop the family of products. Common features are easy to manage due to the reusability in all products that develop from the domain of SPL [?, ?, ?]. Variability features create differentiation and diversification in products and are selected according to end user of requirements [?, ?]. Therefore, variable features are not part of every product and need to manage the relationship with other features during selection and rejection from actual product development. During product configurations selection or rejection of a feature may cause the relationship constraints and leads to invalid product [?, ?]. Therefore, variability management of SPL is the key challenge during the development of the family of software. To enable the high reusability of SPL features, variable features need to handle and manage in efficient way as SPL supports the high reusability of features [?, ?].

To manage the common and variable features of SPL, a tree structure known as Feature Model is used in literature [?, ?, ?]. Feature modeling is a method for the documentation of variable features in an SPL, how the variation points affect each other and what are the rules for establishing the configuration of SPL. Feature model is a compact picture of complete SPL with constraints and relationships among features. Each feature in feature model represents the functionality of product. For the development of unique product of SPL, features are selected from the feature model. During the selection of features from feature model, parent-child relationships must be followed for valid product configuration. Predefined relationships of feature model are, i) alternative features, where one and only one feature is selected from a group of features, ii) optional features, may or may not be selected and iii) OR group, at least one feature must be selected among features from a group [?, ?]. As the feature model is the representation of SPL domain, therefore, all features need to be developed in advance without

developing any running application at the domain level for the reusability in actual product configurations.

For the product configuration from feature model, organizations invest the effort, cost, time etc. in advance to construct the features. [?, ?, ?, ?]. Figure ?? [?] shows the initial cost of SPL and single product development that indicates the SPL organizations invest initial development cost without any market benefits. Therefore, the adaptation of SPL for any organization is based on the total initial development cost to estimate their own budget to develop the family of products. Therefore, considering initial budget, organizations estimate the total cost and then take the decision whether they should adopt the specific SPL or not [?, ?, ?]. The initial development cost of SPL can be estimated by finding the total number of products.

The total number of products is calculated from the feature model by using cardinality relationships (alternative, optional, Or group). Evaluating the total number of products is simple and easy from the single level of relationship (cardinality constraints) feature models. When these relationships has become complex i.e. nested relationship, it also become hard to calculate the total number of products due to chances of relationship constraint violations. Therefore, in complex feature models, where nested cardinality relationships become more complex, calculation the total number of products is a challenging task especially in large feature models where millions of products exist. Furthermore, in simple feature models, valid product configurations and development of product according to market segments are easy due to the simple relationship. However, in complex feature models, it is a challenging and complex task to evaluate each product to estimate the initial development cost and market benefits due to nested relationships. Furthermore, From a large number of product configurations, developers need to select the product configuration according the end user and market

segments. The final product must meet relationship constraints and the end-user specifications for the valid product configuration. In literature, proposed approaches such as Indicator-Based Evolutionary Algorithm (IBEA) [?] and Non-dominated Sorting Genetic Algorithm (NSGA-II) are unable to provide the best solution that meets the 100% correctness i.e. relationship constraints and satisfy end-user requirements from the large and complex feature models. Therefore, from the millions of products, it is a challenging and time-consuming process to select the best solution i.e. 100% correctness of optimum configuration that satisfies the market segments.

To overcome the challenges and problems discussed above, we propose two approaches, the first approach enables to find the total number of products to estimate the initial cost of SPL and evaluate all the valid possible configurations. The second approach enables to find optimum configuration selection that satisfies the market segments.

First, to calculate the total number of products from complex feature models, we propose Binary Pattern for Nested Cardinality Constraints (BPNCC) approach. BPNCC approach enables to counts the total number of products and generates binary patterns for each product from complex feature model. To calculate the total number of products and valid configurations, BPNCC effectively manages the variability of feature model by converting the cardinality relationships of each parent-child relationship (alternative, optional, OR) into the binary pattern (0s, 1s), where 1 indicates selection of feature and 0 represents the rejection of feature. Furthermore, BPNCC approach also determines the valid combinations of features according to nested cardinality constraints of feature model. In BPNCC approach, variability feature model of binary combinations for selected and non-selected features are based on cardinality constraints, such as alternative, optional, and OR group. The beauty of this approach is independent of any tool and

also does not hide the internal information of selected and non-selected features. In Order to show the correctness and effectiveness, we have applied the proposed approach to small and large feature models.

Second, we propose Multi-Objective Optimum (MOO)-BPNCC approach to get the goal-base and minimized, maximized optimum solution for application development without any relationship constraint violation. MOO-BPNCC approach is an extension of BPNCC approach and further consists of two independent and alternative methods to optimize feature model: the first method is suitable to get the goal-base optimization, it applies objective functions on all configurations for optimum solutions. However, this method increases space and execution time on large feature models where millions of product configurations exist. The second method is suitable to get the minimized and maximized optimum solutions by removing optional features that have constant values; 0 for minimization and 1 for maximization of objective functions. For minimized or maximized optimum solutions, execution time and space of feature model can be reduced. In BPNCC approach, we compute all possible solutions without any constraint violations; therefore, there is no possibility to miss any valuable solution for optimum combinations. In this study, we have found the minimized optimum solutions based on four minimized objective functions. We evaluated the outcomes of the goal-based method and minimized, maximized method. Minimized or maximized method is giving the best performance in context of execution time and space. Furthermore, we have performed the experimental comparison of MOO-BPNCC approach with well-known optimization algorithm IBEA from literature and concluded that MOO-BPNCC approach performs better to find the minimized and maximized optimum solutions. To evaluate the effectiveness and efficiency, we have compared MOO-BPNCC with IBEA. MOO-BPNCC finds the 100% correctness of optimum configurations, whereas, IBEA performed

maximum 96% correctness of optimization of feature model.

The main contributions of this dissertation are given as:

- For establishing the domain of SPL, organizations need to estimate the initial cost of SPL. Initial cost of SPL can be estimated by using the total number of products. Organizations can find the total number of products without missing any product by using our proposed BPNCC approach. Furthermore, for product configurations, the adaptation of BPNCC approach is simple and effective for the organizations to find the all possible product configurations of feature model in the binary pattern. The binary pattern of SPL configurations enables to calculate the cost of each product. Before the development of any individual product, organizations can estimate the initial cost.

- By using MOO-BPNCC approach, organizations can find minimized or maximized optimum solutions without any relationship constraint violation for the product development according to market segments. MOO-BPNCC approach also enables to find the goal-base optimum solution that satisfies the end-user objective functions.

The remaining dissertation is organized as follows. Chapter II provides a brief background related to Software Product Line, Feature model and Optimization. Chapter III discusses related work. Chapter IV presents Product configurations of feature model with BPNCC proposed method and also presents the MOO-BPNCC for multi-objective optimization of feature model by using BPNCC. Chapter V provides the brief experimental comparison of MOO-BPNCC with existing optimization algorithms. In the last, Chapter VI concludes dissertation with a summary of the dissertation and future works.

Chapter 2.Background

In this chapter, we present an overview of software product line, feature model, feature-oriented domain analysis, feature-oriented reuse method, attribute feature model and optimization.

2.1 Software Product Line

SPL consists of the set of products that share common and variable features to satisfy the particular market requirements [?] as shown in fig. ??. Organizations adopt the SPL to acquire the business objectives such as time-to-market development, low cost and increase the productivity [?, ?].

At the initial stage of SPL adoption, organizations need to specify their business objectives and scope of products to specify the common features that will be used as the core of every product and variation points that will differentiate each product of SPL [?, ?, ?]. Processes of SPL development are:

- Domain Analysis and Engineering: This process is substitute a complete scope of specific SPL where organization finds the number of products before adoption to predict the development cost and benefits [?, ?]. In each product, the organization needs to identify the features that are required by the end user with different variation points [?].

- Domain Implementation: This is domain development process where all features are developed independently for the derivation of products [?].

- Requirement Analysis: This process is for end-user requirements analysis. End-user specifies the functional (features to be selected) and non-functional (cost and efficient etc.) requirements for the product development. These requirements must be from the scope of SPL domain [?, ?].

- Product Derivation: This process enables the final product derivation according to requirements of the end user. End-user selects the features from the domain that need to be part of product and developers to generate the product [?, ?]. Developers select the features and reuse them for every product derivation. Reusability of features make the product derivation cost-effective and time-to-market [?].

Two main processes of SPL are domain engineering and application engineering.

2.1.1 Domain Engineering

SPL domain engineering creates a platform where all features can be reused for the development of products [?] as shown in fig. ?? [?]. Reusable features are consist of common and variable features. Common features are always part of every product and variable features are used by the requirements of the end-user [?, ?, ?]. Therefore, the development of the SPL platform required the software development artifacts such as requirements, architecture, design, tests etc [?].

There are three main goals of domain engineering process, 1) define the common and variable features of SPL, 2) define the set of products that are

planned to develop i.e. scope of SPL and 3) develop all reusable features that accomplish the required variability. Further, domain Engineers refine and give feedback about the variation points that are determined during the initial stage of domain engineering. There are five sub-processes of domain engineering.

Product Management

This process deals with the market strategy with economic aspects of SPL i.e. allocation of initial development cost, market benefits etc. The product portfolio of business unit is managed by the project management team. Product management team identifies about the compact scope of SPL i.e. what is under the scope and what is outside the scope.

Top management defines the goals of the company and gives input to project management. Further, the product management defines the road-map of SPL according to the variable and common features for the development of future products and the schedule plan for the release of the final product. Furthermore, project management defines the list products with variation points and artifacts that can be reused for the platform development.

Project management of SPL differs from the single system product management due to the platform development that accomplishes the changes in features, constraints and set the standards for future product development from the same platform.

Domain Requirements Engineering

Requirement elicitation and documenting them according to variable and common requirements of SPL is accomplished by the process of domain requirement engineering. The input of this process is the road-map that is deter-

mined by product management. The output contains reusable artifact, model-based and textual requirements and, in particular, the variability model of SPL. Furthermore, this process defines the attribute values for each feature. Therefore, the output does not contain requirements specification of the single product but common and variable functional and non-functional requirements of whole SPL.

In this process, all requirements are specified according to common and variable features, common features must be part of every product and variable features for specific products (i.e. differentiate among several products).

Domain Design

This process encompasses all events for the development of reference architecture of SPL. Reference architecture defines the common and high-level abstraction for all SPL configurations. Domain requirement process provides the input of all common and variable features, rules, reusable artifacts and relationship between features. The output of this process provides the common reference architecture that consists of variability model i.e. SPL internal variability for all configuration but not specific for one product.

In the single system, domain design indicates the single system, however, SPL domain design provides the reference architecture for all products that shares common and variable features i.e. family of software. Due to high flexibility, reference architecture can be reusable according to specific application requirement.

Domain Realization

This process provides the comprehensive design and implementation of reusable components. The input of this process is the list of reusable components that

are determined by reference architecture for the development of products. The output is the complete design and implementation of reusable software artifacts.

Domain realization of SPL differs from the single system due to loosely coupled features, configurable components. Each SPL feature is planned, designed in reference architecture and implemented in the context of reuse in multiple products of SPL.

Domain Testing

Reusable components validate and verified by domain testing. This process tests the artifacts according to specifications (requirements), reference architecture and design components. Furthermore, reusable test artifacts developed by this process to reduce the effort of testing at application development.

Domain testing takes the input from domain requirements, domain design (reference architecture) domain realization (interface designs) and domain realization (designed and implementation of reusable components). The output defines the testing results according to the specification of all variable features. Domain testing does not apply on running application, however, project management provides an application for testing phase to check whether all relationships and rules are not violated. Testing starts from the single component to the chunks of common components that are integrated as dummy application.

2.1.2 Application Engineering

This process is responsible for deriving product line applications from the platform established in domain engineering [?, ?] as shown in fig. ?? [?]. It exploits the variability of product line and ensures the correct binding of the

variable features according to the application's specific needs [?].

Main goals of this process are [?, ?]:

- Enhance the reusability of domain asset artifacts during the development of a product.
- Utilizing the variable and common artifacts of SPL efficient way to increase the reusability during the product development.
- Manage the application artifact according to domain engineering i.e requirements, design, components, test and implementation.
- Manage the variability from domain engineering according to end-user requirements for application development.
- identify the variation points for all products in the scope of SPL domain engineering.

Application Requirement Engineering

This process gathers all the requirements that are required for the specific application development. Application requirements highly depend on the domain engineering requirements due to reuse of components and predefined scope of SPL. Therefore, the main concern is to detect the similarity of application requirements and domain engineering process for the high reusability of domain components. Domain requirement engineering provides the input of SPL scope and application engineering refines that requirements according to application specification that needs to be developed. Therefore, the output of this process is the requirements that are specified by end-user or market.

Requirement elicitation team do not gather the requirements from scratch, end-users select the common and variable features from the domain according to relationship and constraints between features. If some concerned

requirements are new from end-user then this process find the similarity of application requirements and domain requirements to calculate the effort. Later, these new requirements also become part of the domain to increase the reusability for next application development.

Application Design

This process provides the application architecture according to requirements from end-user and domain architecture. Reference architecture from the domain is the base of application architecture. Application architecture selects (variable components) the artifacts from reference architecture according to application specification and generates the architecture that meets the requirement specification of the desired application.

Application design takes the input from application requirement engineering and reference architecture to generate application specific architecture. This architecture indicates the final application requirements and specifications. The development of application architecture is not from the scratch, but develop it from reference architecture with the variable components. Variable components from domain create the difference between all application architectures. Therefore, application designers must be careful about the relationships that are defined by domain design in reference architecture for correct application development.

Application Realization

Reusable components need to identify in this process for the application development. As the reusable components are already assembled at domain realization process, therefore, it generates running application after assembling all components (common and variable).

Application Testing

This process validates and verifies the specifications of running application according to end-user requirements and domain level relationships between components. The inputs of this process are requirements from end-user, the relationship between components and reusable test suits that are defined at domain testing.

2.2 Feature Model

Feature model is tree structure of all domain artifacts such as common and variable features. Common features are mandatory i.e. must be part of all SPL products [?, ?]. However, variable features are selected according to end-user requirements and specification of products. Feature model defines the pre-defined constraints and relationships among features [?]. Therefore, feature model is the compact picture of complete SPL scope where all features and their specifications need to be defined. Basic relationships between variable features are alternative, Or and optional [?, ?] as shown in fig. ??.

- Alternative: Only one feature need to be select i.e. one-to-one.
- Or Group: More than one features can be select i.e. one-to-many.
- Optional: Not necessary to be part of the product.

Furthermore, the extension of these relationships are defined as:

- Mandatory alternative: Only one feature must be select i.e. one-to-one.
- Optional alternative: Not necessary to be part of the product. However, if needs to be select then only one feature can be part of the product

i.e. zero-to-one.

- Mandatory Or group: At least one feature must be part of product i.e. one-to-many.

- Optional Or group: Not necessary to be part of product i.e. zero-to-many.

Comprehensive analysis and information of feature model are essential for assets development of the product line, however, this information is not enough for product derivation [?]. Information related to binding features such as timing and relationship to include in specific products and delivered to stakeholders also the derivation of components in SPL [?]. Moreover, in product line assets, some features (bound features) must be provided at the time of development while, some of them (optional features) can be provided on the time of installation by the choice of costumer [?, ?]. These existing sets of features called as binding units (variability and commonalty units) which further utilized to identify the points of variation to achieve composition components easily for feature bindings. The timing units of feature binding are also influenced component design [?]. In case of when a feature (timing), the load table approach can be adopted for the component design, at the time of installation. It is, therefore, the components need to be designed for the provision of binding features when they are required.

The terminal features are required to bind the derivation of new product from feature model by finding their combination [?]. It can be assumed that the development of a new product is successful if it fulfills all constraints which are defined by the feature model. The constraints which are governed by a feature model on terminal features are “And” (select all), “Or” (select few or all) and “Alternative” (select only one) groups [?, ?].

2.2.1 Feature Oriented Domain Analysis (FODA)

FODA method proposed by Kyo C Kang in 1990, applied on domain engineering to model the domain artifacts into feature model [?]. Requirement analysis for specific product development is call domain analysis. It applies to the family of products rather than on single product development [?]. Effective domain analysis according to feature relationships and constraints enable the product development easy efficient on all products of SPL. It parameterizes the features for all products in the scope of SPL [?, ?]. Domain analysis phases are shown in fig. ??.

FODA supports the development of the family of software in a way:

- Understanding the domain i.e. common and variable features and their binding relationship at the domain level.
- Specifications of features for the development of applications i.e. requirement and scope of SPL.
- Built the reusable resources such as design, domain realization, domain test suits etc.
- Assisting the SPL processes for domain analysis creation and other reusable components.

2.2.2 Feature Oriented Reuse Method (FORM)

FORM method proposed by Kyo C Kang in 1998 is used to analyze the common and variable features to capture the differences of applications in domain engineering according to features combination [?]. Moreover, FORM supports the development of domain architecture and features by using the analysis results to increase the reusability [?]. Commonalities and variabilities are captured by feature model and reuse the domain architecture and

developed features for the application development. By defining the features in terms of common and variable in domain engineering, reusability increased in application development [?].

2.2.3 Attribute Feature Model

Attribute feature model is the extension of feature model where all features are defined by the attribute values. In attribute feature model, every feature contains functional and non-functional quality attributes [?, ?]. Functional and non-functional requirements show the specification of every feature and describe the importance of application development. End-user selects the features for application development by analyzing the quality attributes of every feature [?]. Moreover, end-user enables to perceive the efficiency of application in advance according to requirements. Based on quality attributes, features are selected for product derivation according to the user requirements [?]. Figure ?? shows the attribute feature model with four quality attributes: cost, performance, CPU and memory.

2.3 Optimization

Optimization is the process to find optimum solutions from the set of solutions. In academia and software industry, it is common problem to get the optimum solutions according to different market requirements due to constraints and relationships among factors [?, ?]. To find the optimal solutions according to constraints and relationships, multiple heuristic based algorithms have been proposed such as genetic algorithm, NSGA, IBEA etc [?]. These are search base algorithms that evaluate the fitness (quality attributes) value of every possible solution according to given objective functions and compare them to find the non-dominated (minimized or maximized) solutions. MOEAs are commonly used to find the multi-objective optimum solutions in

the software industry to search with respect to cost and performance etc. [?].

국문요지

전직대통령의 신분과 예우에 관하여는 법률로 정한다. 국가는 농업 및 어업을 보호·육성하기 위하여 농·어촌종합개발과 그 지원등 필요한 계획을 수립·시행하여야 한다.

대법원장과 대법관이 아닌 법관의 임기는 10년으로 하며, 법률이 정하는 바에 의하여 연임할 수 있다. 원장은 국회의 동의를 얻어 대통령이 임명하고, 그 임기는 4년으로 하며, 1차에 한하여 중임할 수 있다.

대통령후보자가 1인일 때에는 그 득표수가 선거권자 총수의 3분의 1 이상이 아니면 대통령으로 당선될 수 없다. 감사원은 세입·세출의 결산을 매년 검사하여 대통령과 차년도국회에 그 결과를 보고하여야 한다.

국민의 모든 자유와 권리는 국가안전보장·질서유지 또는 공공복리를 위하여 필요한 경우에 한하여 법률로써 제한할 수 있으며, 제한하는 경우에도 자유와 권리의 본질적인 내용을 침해할 수 없다.

대통령이 제1항의 기간내에 공포나 재의의 요구를 하지 아니한 때에도 그 법률안은 법률로서 확정된다. 법관은 탄핵 또는 금고 이상의 형의 선고에 의하지 아니하고는 파면되지 아니하며, 징계처분에 의하지 아니하고는 정직·감봉 기타 불리한 처분을 받지 아니한다.

Acknowledgments

계엄을 선포한 때에는 대통령은 지체없이 국회에 통고하여야 한다. 국교는 인정되지 아니하며, 종교와 정치는 분리된다. 일반사면을 명하려면 국회의 동의를 얻어야 한다.

헌법재판소의 조직과 운영 기타 필요한 사항은 법률로 정한다. 공무원의 신분과 정치적 중립성은 법률이 정하는 바에 의하여 보장된다. 대통령의 임기가 만료되는 때에는 임기만료 70일 내지 40일전에 후임자를 선거한다.

헌법재판소는 법률에 저촉되지 아니하는 범위안에서 심판에 관한 절차, 내부규율과 사무처리에 관한 규칙을 제정할 수 있다. 국회의원은 국회에서 직무상 행한 발언과 표결에 관하여 국회외에서 책임을 지지 아니한다.

법관은 헌법과 법률에 의하여 그 양심에 따라 독립하여 심판한다. 국군의 조직과 편성은 법률로 정한다. 모든 국민은 양심의 자유를 가진다. 모든 국민은 법 앞에 평등하다. 누구든지 성별·종교 또는 사회적 신분에 의하여 정치적·경제적·사회적·문화적 생활의 모든 영역에 있어서 차별을 받지 아니한다.

형사피의자 또는 형사피고인으로서 구금되었던 자가 법률이 정하는 불기소처분을 받거나 무죄판결을 받은 때에는 법률이 정하는 바에 의하여 국가에 정당한 보상을 청구할 수 있다.