# Merancang Perangkat Lunak Sistem Penjaminan Mutu Internal (SPMI) Perguruan Tinggi yang Memiliki Daya Adaptasi Terhadap Perubahan Kebutuhan Pengguna secara Cepat dan Sering

Acep Taryana[1], Ari Fadli[1], Siti Rahmah Nurshiami[2]

[1]Teknik Elektro, Fakultas Teknik, Universitas Jenderal Soedirman,
Jalan Profesor DR. HR Boenyamin No 708, Purwokerto 53122
[2]Matematika, Fakultas MIPA, Universitas Jenderal Soedirman,
Jalan Profesor DR. HR. Boenyamin No 708, Purwokerto 53122

Penulis untuk Korespondensi/E-mail: acep@unsoed.ac.id

*Abstrak* – **Salah satu permasalahan mendasar proses pengembangan perangkat lunak adalah kebutuhan yang tidak tertangkap lengkap saat awal pengembangan, atau abstraksi kebutuhan pengguna yang kurang terpetakan secara sistematis, runut oleh pengembang. Ketidaksempurnaan pengungkapan kebutuhan pengguna tersebut dapat mengakibatkan produk perangkat lunak yang kurang lengkap bahkan bisa tidak sesuai dengan kebutuhan pengguna. Ketidaksesuaian baru dapat diamati oleh pengguna setelah pengembang menyelesaikan setiap rilis produk. Pada paper ini akan ditunjukkan bagaimana sebuah perangkat lunak SPMI dirancang, diterima oleh pengguna, diberikan masukan oleh pengguna atas masukan perbaikan pada kurun waktu tertentu. Metode yang digunakan dalam proses pengembangan perangkat lunak adalah metode DevOps yang memiliki kemampuan untuk mensinkronkan kebutuhan pengguna dengan pengembangan aplikasi yang berkelanjutan, cepat selama pengembangan dan pengoperasian berlangsung. Metode DevOps tidak hanya mengelola bagian pengembangan tetapi juga mengelola bagian pengoperasian. Hasil perancangan menunjukkan bahwa DevOps menjadi pendekatan tepat agar perangkat lunak pengembangan SPMI dikembangkan dari kecil menjadi besar, *step by step* tetapi tanpa kehilangan penelusuran antara rilis produk. Dan yang lebih mendasar, DevOps mampu memperkecil gap antara pengembang dengan pengguna aplikasi SPMI-PT. Melalui metode DevOps, pengembangan dan pengoperasian memiliki keterhubungan sebagai timbal balik antara pengembangan dengan pengoperasian maupun sebaliknya.**

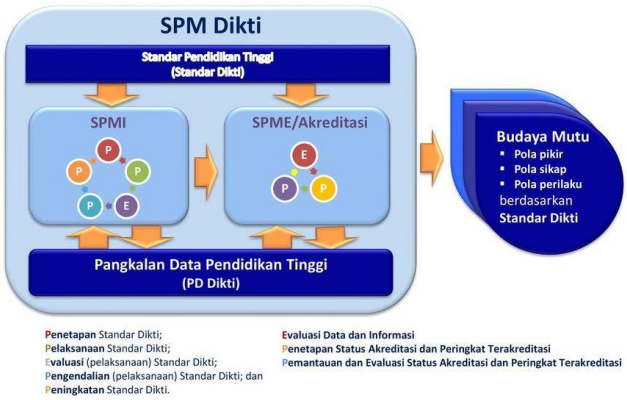
*Abstract* – **One of the fundamental problems of the software development process is the requirement that is not completely captured at the beginning of development, or the abstraction of user requirements that are less systematically mapped out, by the developer. Imperfections in disclosing the requirements of these users can result in incomplete software products that may not even match the user's needs. New nonconformities can be observed by users after the Developer completes each product release. In this paper, we will show how an SPMI software is designed, accepted by the user, given input by the user for input improvement in a certain period time. The method used in the software development process is the DevOps method which can synchronize user requirements with rapid, rapid application development during development and operation. The DevOps method not only manages the development part but also manages the operation part. The design results show that DevOps is the right approach so that SPMI development software is developed from small to large, step by step but without losing traces between product releases. And more fundamentally, DevOps can reduce the gap between developers and SPMI-PT application users. Through the DevOps method, development and operation have a relationship as a trade-off between development and operations and vice versa.**

***Keywords*** *– Software engineering, Devops, Continuous improvement*

## PENDAHULUAN

Sistem penjaminan mutu perguruan tinggi merupakan suatu rangkaian proses yang terdiri dari perencanaan, pemenuhan, pengendalian, dan pengembangan standar pendidikan tinggi yang dilakukan secara konsisten dan berkelanjutan, sehingga pemangku kepentingan *(stakeholders)* internal dan eksternal perguruan tinggi, yaitu mahasiswa, dosen, karyawan, masyarakat, dunia usaha, asosiasi profesi, pemerintah memperoleh kepuasan atas kinerja dan keluaran perguruan tinggi. Hadirnya kegiatan penjaminan mutu ini merupakan perwujudan akuntabilitas dan transparansi pengelolaan perguruan tinggi.

Sesuai Permenristekdikti No.62 Tahun 2016, menyatakan bahwa Sistem Penjaminan Mutu Pendidikan Tinggi (SPMPT) terdiri atas Sistem Penjaminan Mutu Internal (SPMI) dan Sistem Penjaminan Mutu Eksternal (SPME). Kedua jenis penjaminan mutu tersebut membutuhkan ketersediaan Pangkalan Data Perguruan Tinggi (PDPT) Nasional sebagai sumber data melaksanakan penjaminan sesuai standar Dikti, sebagaimana tertuang seperti pada Gambar 1.




Gambar 1. Sistem Penjaminan Mutu PT [1]

Berdasarkan hal tersebut peneliti terinspirasi untuk melakukan penelitian terkait mendeskripsikan proses perencanaan, implementasi, serta kendala yang dihadapi dalam pelaksanaan sistem penjaminan mutu internal secara keseluruhan, proses evaluasi serta pemanfaatan hasil implementasi sistem penjaminan mutu internal dalam rangka meningkatakan mutu pendidikan secara berkelanjutan berbasis pada konsep pengembangan sistem informasi di lingkungan kerja peneliti.

Menurut Lestarirevolusi industri 4.0 tidak hanya berdampak pada dunia perindustrian saja, namun juga pendidikan tinggi. Pola pikir revolusi industri 4.0 perlu diterapkan dalam pengelolaan pendidikan tinggi baik dalam proses bisnis, kurikulum, pembelajaran maupun riset. Dalam artikel dijelaskan bahwa ICT pendidikan tinggi semestinya harus berciri 5.0 di saat industri baru memasuki 4.0, hal ini membuktikan bahwa inovasi pengelolaan ICT pendidikan tinggi sangat penting, karena menjadi tonggak kemajuan teknologi perindustrian [2]

Pemberlakuan PDPT untuk menunjang pengukuran mutu PT sudah dinyatakan oleh Ristekdikti, namun mekanisme/model untuk selalu menyediakan data yang sesuai dengan kebutuhan SPMI-PT pada setiap siklusnya belum ada yang mengupas atau membahasnya. Berdasarkan kondisi saat ini ada dua alternatif untuk mencapai universitas yang bermutu, yaitu 1). Mengembangkan Sistem Informasi (SI) berdasarkan kebutuhan SPMI-PT, 2). Pengembangan SPMI-PT berdasarkan ketersediaan data dalam basisdata SI.

Kelemahannya, kedua alternatif tersebut hanya cocok untuk implementasi SPMI-PT/SI, khusus untuk standar mutu (indikator) yang bersifat umum dan telah ditetapkan sebelum pengembangan SI/SPMI-PT. Sedangkan bila ada indikator baru sebagai masukan baru dari para pengguna (fakultas, jurusan, program studi) yang diperlukan saat implementasi berjalan, kedua alternatif itu akan mengalami kesulitan untuk melakukan perbaikan secara berkelanjutan dan otomatis. Terlebih jika ada tuntutan perubahan SI, perubahan format data akan membutuhkan usaha yang besar untuk mewujudkannya.

Selain hal tersebut, hasil pengamatan peneliti pada implementasi Sistem Informasi (SI) dan Sistem Penjaminan Mutu (SPMI-PT) di lingkungan kerja peneliti sejak tahun 2005 sampai dengan sekarang. Implementasi SI dan SPMI-PT masih dilakukan secara terpisah seperti tidak ada keterhubungan. Sehingga peneliti berkeinginan untuk memadukan pengembangan Sistem Informasi dan Sistem Penjaminan Mutu Internal dalam rangka meningkatakan sistem penjaminan mutu yang berkelanjutan.

Oleh karena itu, bagaimana merancang sebuah perangkat lunak yang memiliki adaptasi tinggi terhadap berbagai perubahan, mencakup perubahan struktur data, perubahan kebutuhan infrastruktur, dan lainnya. Dan yang menjadi fokus pada penulisan paper ini adalah bagaimana membangun

sebuah perangkat lunak yang memperhatikan perencanaan pengembangan, perancangan, pemrograman, pengujian, pemasangan dan penyebaran perangkat lunak dalam infrastruktur target sehingga pengembangan memperhatikan 2 sisi yaitu sisi *Development* dan *Operation*.

Pada pendekatan pengembangan *perangkat* lunak yang lalu seperti metode *Waterfall, Prototyping, Agile*, RUP, hanya membahas saat pengembangan perangkat lunak saja, tidak ada topik untuk membahas bagaimana menyebarkan, memasang perangkat lunak dalam infrastruktur target. Padahal kondisi saat ini menuntut bahwa pengembangan harus tuntas sampai dengan penyebaran dan pemeliharaan rilis perangkat lunak. Salah satu kelemahan pengembangan perangkat lunak pada sebuah organisasi tradisional yang sering terjadi adalah ketika pengembangan aplikasi yang dilakukan oleh tim *development* dan tim *operations*, memiliki tujuan berbeda [3]. Situasi yang dapat ditangkap dari sini adalah bahwa pada tim *development* bertugas menciptakan fungsi baru dan memperbaiki *bug*, sementara itu tim operations akanmembuat infrastruktur yang handal yang memungkinkan aplikasi berjalan dengan stabil. Dapat disimpulkan, peran *development* dan *operations* sangat bertentangan, *development* berusaha untuk selalu melakukan perubahan sedangkan *operations* berusaha untuk mempertahankan kestabilan [8]. Hal ini menjadi masalah ketika merilis perangkat lunak untuk waktu yang cepat dan sering, karena keduanya sangat bergantung satu sama lain. Oleh karena itu, budaya tradisional dinilai kurang efektif untuk pengembangan aplikasi di masa sekarang. Berdasarkan penelitian [4], kerjasama antara tim IT dalam pengembangan dan pemeliharaan perangkat lunak multak diperlukan.

Pendekatan pengembangan perangkat lunak yang lama masih menimbulkan *gap* di antara *development* dengan *operation*. Oleh karena itu kami mengusung pendekatan baru yaitu pendekatan DevOps sebagai salah satu alternative mengantisipasi kelemahan penggungaan metode lama. DevOps merupakan paradigma yang muncul untuk menghilangkan batas dan penghalang antara developer dan petugas pengoperasi yang secara tradisional ada dalam banyak perusahaan saat ini. Tujuan utama dari DevOps adalah untuk dapat melaksanakan pengembangan dan pengiriman perangkat lunak yang terus menerus ("*continuous delivery*") agar muncul rilis yang lebih cepat dan sering. Hal ini memungkinkan dapat membuat respon yang cepat untuk mengantisipasi perubahan kebutuhan pengguna dan akan menjadikan keuntungan kompetitif yang kritis [9]. Beberapa peneliti menyebut adanya budaya DevOps. Budaya tersebut dapat mengaburkan batas antara peran development dan operations dan akhirnya dapat menghilangkan perbedaan [5].Sikap tanggung jawab bersama merupakan salah satu aspek budaya DevOps yang mendorong kolaborasi yang lebih erat. Sehingga tidak ada batas antara *development* dan *operations*. Pengadopsian DevOps di beberapa perusahaan maupun organisasi menunjukkan manfaat bagi pengembangan produk aplikasi mereka, seperti yang ditunjukkan pada penelitian Ellen [6]. Devops mendukung kecepatan produksi perangkat lunak dan otomasi, mengurangi usaha yang dibutuhkan ketika menyiapkan perilisan, sehingga memungkinkan organisasi untuk rilis lebih sering sesuai keperluan [6][7].

Penelitian dasar tentang DevOps di dunia internasional sudah muncul sejak 10 tahun yang lalu seperti pada Tabel 1. Berbagai penelitian telah disusun mulai dari peneliti yang membuat istilah DevOps, konsep DevOps dalam layanan Cloud, memodelkan user dan customer sebagai sebuah siklus umpan balik dalam DevOps, DevOps sebagai sarana kolaborasi dan otomatisasi sebuah system. Bahkan dalam paper ini kami tunjukkan pula tentang literatur penggunaan istilah rantai pasok dalam dunia manajemen industri. Memperhatikan istilah rantai pasok pada Tabel 1, sangatlah tepat digunakan untuk menyatakan bahwa DevOps menyerupai rantai pasok dimana ada customer dan user yang saling berkolaborasi. DevOps sebagai jalur penyebaran (delivery pipeline).Istilah lain yang sangat penting adalah "Continous Delivery", sebagai keunggulan manajemen untuk menyampaikan produk kepada customer secara cepat dan sering.

Lebih spesifik, paper ini memiliki dampak terhadap pencarian model perancangan yang mendekati kebutuhan pengguna. Namun merujuk pada penelitian sebelumnya [15], pencarian model rancangan perangkat lunak dilakukan secara *tryanderror* melalui berbagai rilis perangkat lunak, tidak melalui pendekatan eksak seperti menggunakan metode Formal. Metode formal berfungsi untuk memodelkan kebutuhan pengguna agar mendapatkan ketepatan perancangan (design) yang kemudian berpengaruh pula pada ketepatan pemrograman yang sesuai dengan kebutuhan.

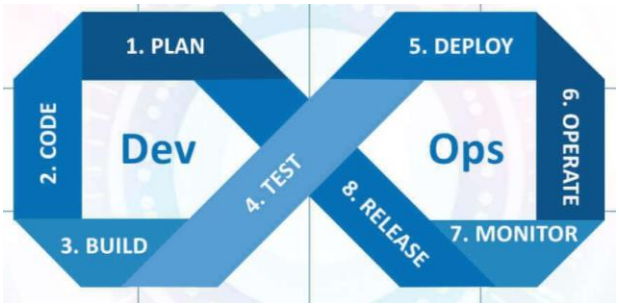
Tabel 1. Penelitian Sebelumnya

| No | Referensi | Latar belakang/Temuan/Pembahasan |
|---|---|---|
| 1 | [11], [12] | Portabilitas dan pengaturan terotomatisasi aplikasi perusahaan (*Enterprise*) merupakan masalah utama pengembangan IT saat ini. Layanan *Cloud* dapat dibuat portabel, manajemen juga harus dapat dibuat portabel untuk lingkungan target. Pada paper ini penulis memperlihatkan bagaimana perencanaan *Topology and Orchestration Specification for Cloud Applications* (TOSCA) dapat mengaktikan portabilitas dari aspek-aspek operasional. |
| 2 | [8] | User dan **customer** aplikasi *web* dan *mobile* mengharapkan umpanbalik yang cepat terhadap permasalahan dan permintaan kebutuhan. Merupakan sebuah keuntungan yang sangat kritis untuk dapat merespon dengan cepat. Disamping perubahan organisasi dan kultur, juga penting untuk mengimplementasikan **DevOps** dalam tataran praktis, peralatan dibutuhkan untuk mengimplementasikan otomatisasi *end-to-end* proses penyebaran (***deployment***). Otomatisasi merupakan kunci untuk mengefisienkan kolaborasi dan mengintegrasikan yang sangat kuat antara *development* dan operasional. Namun perlu usaha keras mengintegrasikan dan mengkombinasikan atomatisasi DevOps Untuk menyebarkan aplikasi ke dalam lingkungan *cloud*. |
| 3 | [9] | Dalam paper ini dibahas tentang DevOpSlang, merupakan bahasa yang berkonjungsi dengan sebuah metodologi untuk mengimplementasikan **DevOps** sebagai cara efisien untuk tujuan kolaborasi dan otomatisasi. Kolaborasi dan **otomatisasi** yang efisien merupakan kunci penggerak untuk mengimplementasikan ***continuous delivery*** dan kemudian untuk mengantisipasi **perubahan** kebutuhan ***customer*** yang cepat. |
| 4 | [13] | Perusahaan seringkali tidak memiliki konsep topologi perusahaan yang terintegrasi dan komprehensif yang merepresentasikan infrastruktur IT, perangkat lunak, layanan yang dijanjikan maupun yang tidak dijanjikan, proses-proses, beserta **relasinya**. Lebih khususnya karena adanya akuisisi, penggabungan, reorganisasi, dan outsourcing tidak ada '**bigpicture'** yang jelas untuk menggambarkan topologi perusahaan. Oleh karena itu pengaturan aplikasi menjadi sulit dan duplikasi komponen dan sistem informasi menjadi meningkat. Lebih jauhnya lagi, kurangnya pemahaman akan membuat perubahan dalam topologi perusahaan seperti konsolidasi, migrasi atau *outsourching* menjadi lebih komplek dan rawan kesalahan yang memicu biaya operasional tinggi. Dalam paper ini dimodelkan *Enterprise Topology Graphs* (ETG) sebagai model formal untuk menggambarkan topologi perusahaan berbasis teori graf ETG untuk layanan *Cloud*. |
| 5 | [14] | Manajemen rantai pasokan (*supply-chain management*) adalah pengintegrasian aktivitas pengadaan bahan dan pelayanan, pengubahan menjadi barang setengah jadi dan produk akhir, serta pengiriman ke pelanggan. Tujuannya adalah untuk membangun sebuah rantai pemasok yang memusatkan perhatian untuk memaksimalkan nilai bagi pelanggan (***customer***). Kunci bagi manajemen rantai pasokan yang efektif adalah menjadikan para pemasok sebagai "mitra" dalam strategi perusahaan untuk memenuhi **kebutuhan** pasar yang selalu **berubah**. |

## METODE

Metode yang digunakan pada perancangan perangkat lunak SPMI-PT adalah DevOps. Metode DevOps merupakan pendekatan praktis untuk mendapatkan perangkat lunak yang sesuai dengan kebutuhan pengguna. Secara umum, pendekatan DevOps dapat dipandang terdiri dari 2 bagian, yaitu bagian *Development*, dan bagian *Operation*. Apabila digambarkan seperti angka 8 pada Gambar

2, *Development* dan *operation* memiliki porsi masing masing 0,5. Pada umumnya, pengembangan perangkat lunak menggunakan metode lama masih berkutat pada area *Development*. Sehingga apabila dihitung secara total, maka pengembangan baru mencakup lingkup 50% dari seluruh usaha pengembangan dan pemeliharaan perangkat lunak. Yang merupakan area *Development* meliputi tahap *Plan*, *Code*, *Build*, *Test*. Sedangkan area *Operation* meliputi *Deploy*, *Operate*, *Monitor*, *Release*. Area *Development* seterusnya disingkat Dev, sedangkan area *Operation* disingkat dengan Ops. Jika kedua kata tersebut dipadukan maka menjadi sebuah brand yaitu "DevOps".




**Sumber :**https://www.weblineindia.com/wp-content/uploads/2018/05/top-10-devops-tools-1.jpg

Gambar 2. Siklus Pengembangan Perangkat Lunak Berdasarkan Paradigma DevOps [10]

Selanjutnya, bagaimana permasalahan SPMI-PT dapat dikomputerisasi dengan memperhatikan sifat dari penjaminan mutu yaitu perbaikan perkelanjutan (*continuous improvement*), dari kecil menjadi besar, tidak harus menunggu semua indikator mutu terbentuk. Berdasarkan penjelasan DevOps di atas, keduanya memiliki sifat yang mirip. DevOps dan SPMI-PT memiliki siklus yang ke-1, 2, 3, dan ke-n untuk proses perbaikan. Oleh karena itu **tahap pertama** yang sangat penting adalah memodelkan kesepadanan antara siklus DevOps dengan Siklus SPMI-PT, tahapan apa saja yang merupakan kelompok *Dev*, dan mana yang merupakan tahapan *Ops*. **Tahap kedua** adalah menginventarisir perkakas (*tools*) untuk mengimplementasikan masing masing tahapan. **Tahap ketiga** adalah pelaksanaan pengembangan perangkat lunak SPMI-PT. Dalam pengembangan ini seluruh tahapan dalam siklus DevOps harus dilalui untuk mendapatkan perangkat lunak yang mendekati kesempurnaan. Semakin banyak siklus yang dijalankan maka ekuivalen dengan produk perangkat lunak yang semakin baik. Tetapi tentunya perlu diperhatikan masalah *budget* jika semakin banyak siklus yang dijalankan.

## HASIL DAN PEMBAHASAN

Hasil perancangan tahap pertama adalah adanya gambaran kesepadanan antara SPMI-PT dengan siklus DevOps. Pengembangan perangkat lunak SI dan siklus pengembangan SPMI-PT memiliki karakteristik atau pola yang sama sebagai sebuah bidang pekerjaan yang dapat diterapkan otomatisasi kepadanya. Secara umum, tahapan pengembangan perangkat lunak SI meliputi pemodelan bisnis, analisis, perancangan, pemrograman, instalasi, penyerahan produk, pengoperasian, monitoring dan evaluasi. Sedangkan tahapan pengembangan SPMI-PT meliputi penetapan standar, pelaksanaan, monitoring dan evaluasi, audit mutu internal atau evaluasi kolega eksternal. Keduanya memiliki isu yang sama yaitu, *continuous improvement*, *continuousdelivery*, *customer*, kebutuhan yang berubah. Sangatlah penting untuk mengimplementasikan SPMI-PT dengan menerapkan DevOps yang terkait dengan pengembangan perangkat lunak SI, sebagai otomatisasi SPMI-PT ke dalam rantai pasokan perangkat lunak. Pengelompokan dan sudut pandang keduanya berdasarkan siklus DevOps dijelaskan dalam Tabel 2.

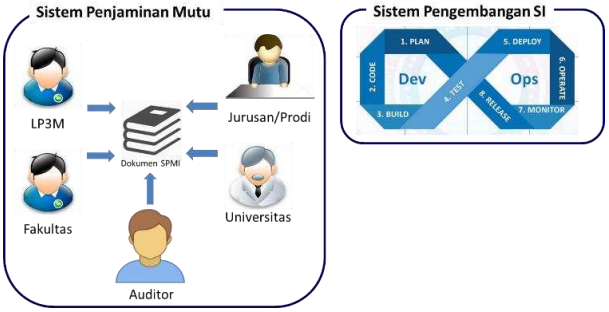
Tabel 2. Pengelompokan tahap pengembangan perangkat lunak SI dan SPMI-PT

| Area | Tahapan Pengembangan Perangkat Lunak SI | Tahapan Pengembangan SPMI-PT |
|---|---|---|
| Area Dev | 1. **Pemodelan Bisnis.**<br>2. **Analisis.**<br>3. **Perancangan.**<br>4. **Pemrograman.**<br>5. **Build**<br>6. **Pengujian** | 1. Penetapan Standar Mutu |
| | **Tahapan Pengembangan Perangkat Lunak SI** | **Tahapan Pengembangan SPMI-PT** |
| Area Operasi | 7. **Instalasi dan Penyerahan produk.**<br>8. **Pengoperasian.**<br>9. **Monitoring**<br>10. **Evaluasi** | 2. Pelaksanaan<br>3. Monitoring dan Evaluasi<br>4. Audit mutu internal atau evaluasi kolega eksternal<br>5. Peningkatan kualitas dan benchmarking. |

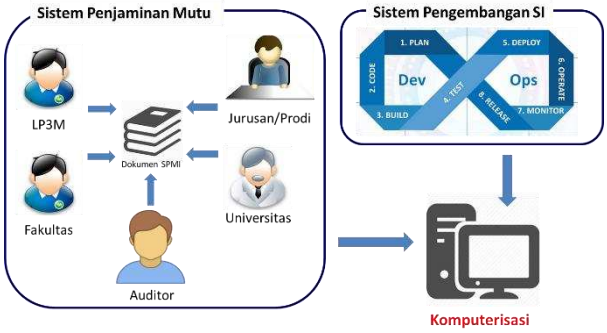
Salah satu keuntungan penerapan DevOps dalam pengembangan perangkat lunak SI adalah memperkecil *gap* antara tim *developer* dengan tim

pengoperasi perangkat lunak SI, yang dapat diperbaiki oleh *developer* secara terus menerus tanpa mengganggu produk yang telah beroperasi. Makna terus menerus berimplikasi pada model pengembangan perangkat lunak SI yaitu pengembangan dapat dilakukan dari hal kecil ke besar dan langsung dapat dipakai oleh *customer*.

Gambar 3 menunjukkan bahwa pengembangan SPMI-PT tidak ada keterhubungan dengan pengembangan perangkat lunak SI. Pada kondisi ini implementasi SPMI-PT dapat dibantu dengan adanya sebuah perangkat lunak SI, namun membutuhkan usaha keras untuk mewujudkannya karena berbagai faktor seperti perbedaan organisasi SPMI-PT dengan SI, komunikasi antara pengembang SI dengan tim SPMI-PT. Sedangkan Gambar 4 menunjukkan pengembangan SPMI-PT dapat dikembangkan secara otomatis dengan melibatkan rantai pasokan perangkat lunak SI menggunakan metode DevOps.



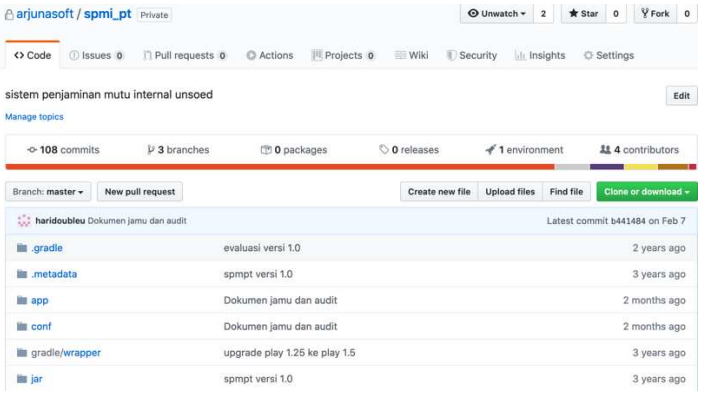

Gambar 3. Sebelum Otomatisasi SPMI-PT diterapkan pada Rantai Pasokan Perangkat Lunak SI.




Gambar 4. Setelah Otomatisasi SPMI-PT diterpakan pada Rantai Pasokan Perangkat Lunak SI.

Karena pengembangan SPMI-PT memiliki kesamaan pola dengan pengembangan perangkat lunak SI, maka peneliti berasumsi bahwa DevOps juga mempengaruhi implementasi SPMI-PT sehingga pengembangan dapat dilakukan secara terus menerus walaupun orang sebagai personil SPMI-PT berganti, bahkan organisasi mengalami perubahan. Dan lebih penting lagi, pengembangan SPMI-PT secara otomatis akan terintegrasi dengan pengembangan perangkat lunak SI. Dua kondisi, sebelum dan sesudah penelitian ini dapat dilihat dalam Gambar 3, Gambar 4

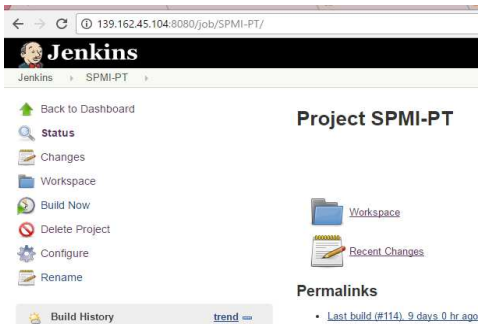
Implemetasi penerapan DevOps dapat diatasi melalui penggunaan berbagai perkakas meliputi perkakas manajemen kolaborasi seperti gitlab, dapat diakses dalam situs www.gitlab.com, github dapat diakses dalam situs www.github.com. Pada penelitian ini digunakan github seperti pada Gambar 5.




Gambar 5. Contoh Penggunaan github

Manajemen kolaborasi berfungsi untuk mengkolaborasikan banyak Developer yang mengembangkan perangkat lunak SPMI-PT pada waktu yang hampir bersamaan tanpa saling menimpa kode sumber, perilisan perangkat lunak juga mudah dilakukan.

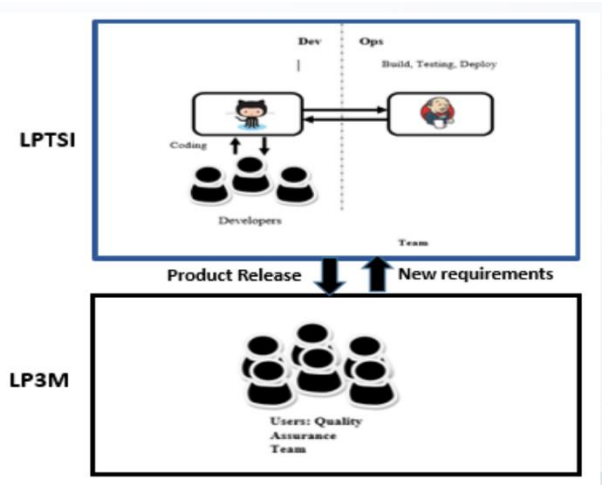
Selain perkakas manajemen kolaborasi, pelaksanaan DevOps memerlukan perkakas untuk manajemen penyampaian produk. Pada penelitian ini kami menggunakan perkakas **Jenkins** untuk menyebarkan aplikasi perangkat lunak dalam server target. Contoh penggunaan Jenkins dapat dilihat pada Gambar 6.




Gambar 6. Manajemen Penyebaran dengan Jenkins

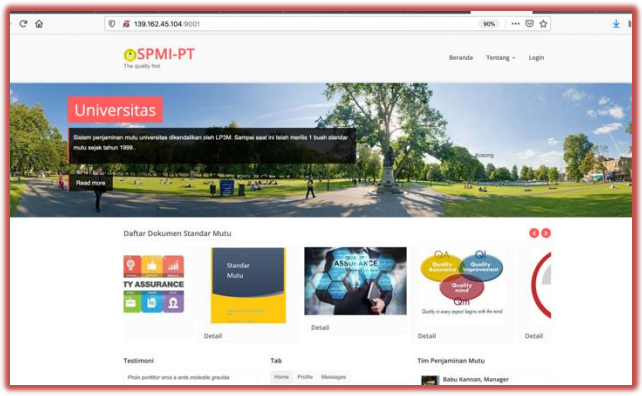
Tim developer memiliki tanggung jawab untuk melaksanakan perencanaan (*plan*), membuat program aplikasi (*code, build, test*), merilis program aplikasi (*release*), dan menyebarkan aplikasi (*deploy*). Sedangkan tim pengoperasi memiliki tanggung jawab untuk mengoperasikan (*operate*), dan melaksanakan monitoring (*monitor*). Setiap tanggung jawab perkerjaan tersebut dikerjakan secara berurutan, mulai dari perencanaan sampai dengan monitoring untuk setiap siklusnya. DevOps yang terotomatisasi mengeksekusi siklus dengan cepat dan sering untuk mengantisipasi perubahan kebutuhan baru agar sesuai dengan kebutuhan *customer*. Penjelasan siklus DevOps dapat dilihat pada Gambar 2.

Hal yang paling penting untuk memadukan pengembangan SPMI-PT dan Perangkat lunaknya adalah melakukan perubahan budaya kerja, dengan cara mengkolaborasikan tim SPMI-PT dengan Pengembangan Perangkat Lunak. Merancang organisasi yang *adapatable* terhadap perubahan baru SPMI-PT, dan cepat ditangani oleh pengembang perangkat lunak. Pada penelitian ini dirancang organisasi antara pengguna dengan pengembang seperti pada Gambar 7.




Gambar 7. Model Organisasi antara Pengembang (LPTSI) dengan Pengguna (LP3M) Perangkat Lunak SPMI-PT

Gambar 7 menjelaskan bahwa unit LPTSI membutuhkan masukan kebutuhan pengguna baru (*new requirements*), dan LP3M membutuhkan Rilis Perangkat lunak baru (*product release*) sesuai permintaan dengan cepat dan sering. Para developer di LPTSI akan dengan mudahnya bekerja secara terkolaborasi atas masukan baru dari pengguna, mudahnya melaksanakan delivery produk baru setelah dirilis ke dalam server target, dan LP3M tidak perlu melihat hasil perubahan ke dalam lingkungan komputasi para Developer. LP3M terkonsentrasi, dapat memonitor hasil pekerjaan setiap saat ke antarmuka yang ditentukan seperti pada Gambar 8.




Gambar 8. Produk perangkat lunak SPMI-PT

Untuk membangun produk perangkat lunak SPMI-PT seperti pada Gambar 8, tentunya belum cukup hanya menggunakan perkakas yang disebutkan di atas, dan belum cukup hanya memikirkan organisasi antara LPTSI dengan LP3M. Isu lain adalah pemilihan teknologi pemrograman yang *adaptable* terhadap perubahan kebutuhan SPMI-PT setiap saat. Pada penelitian ini dipilih Teknologi Java seperti Play Framework1.5 karena mampu menjawab tantangan tersebut. Dan teknologi java Play Framework mampu menerapkan *forward and reverse engineering*. Salah satu hal yang sangat penting menggunakan Play Framework 1.5 adalah Rancangan Class dapat menghasilkan *scheleton source code*, dan mampu menyusun struktur basis data secara otomatis setelah dilaksanakan kompilasi terhadap aplikasi yang dibuat. Pengembang tidak dibebani harus merancang struktur basis data aplikasi karena sudah dibuatkan oleh Play framework. Fasilitas ini mempercepat pengembangan perangkat lunak karena tahap pengembangan database terbantu dengan memanfaatkan pemodelan rancangan *Class*.

## KESIMPULAN

Berdasarkan pada bagian Hasil dan Pembahasan, Penggunaan metode DevOps untuk memenuhi kebutuhan pengguna secara cepat dan sering sangatlah mendukung karena beberapa faktor:

1. Metode DevOps tidak hanya membahas tahap *Development* tetapi juga tahap operasi. Permasalahan operasi seperti pemasangan produk perangkat lunak dilaksanakan terautomatisasi menggunakan perkakas

**Jenkins**. Berbeda jika pembuatan produk menggunakan pendekatan lama, tahap operasi tidak menjadi bahasan. Sehingga saat pemasangan produk pada lingkungan *development* maupun produksi merupakan pekerjaan tersendiri, dan memiliki berbagai permasalahan tersendiri, seperti adanya usaha dari tim operasi yang mempertahankan kestabilan.

2. Metode DevOps sangat adaptasi terhadap pengembangan perangkat lunak SPMI-PT karena adanya sifat/kemiripan yaitu *continuous improvement* dari keduanya. Sifat tersebut sangat mendukung pengembangan perangkat lunak dari kebutuhan yang terkumpul minimalis, tetapi pengguna sudah bisa melihat *prototype* dengan cepat. Dengan ketersediaan *prototype* yang cepat, pengguna akan dengan cepat pula memberikan masukan baru atas produk perangkat lunak yang rilis terakhir.
3. Kedua belah pihak bekerja pada lingkungan masing-masing terhubung dengan infrastruktur teknologi untuk manajemen kolaborasi, manajemen penyebaran perangkat lunak, tidak saling mengganggu, bahkan lebih saling terkolaborasi.
4. Dukungan pemilihan teknologi pemrograman menggunakan paradigma objek, seperti Play Framework, membantu mempercepat pengembangan perangkat lunak SPMI-PT karena ada beberapa tahapan pengembangan yang biasa dilakukan pada metode lama dapat direduksi. Pengembangan rancangan *database* tidak menjadi fokus, cukup merancang *Class* yang kemudian dilakukan generate model *Class*. Model *Class* dalam Play Framework dapat digenerate menjadi database.

Secara umum, Metode DevOps menjadi pendekatan tepat agar perangkat lunak SPMI-PT dikembangkan dari kebutuhan yang terbatas (kecil) menuju membesar, rilis produk sering dilaksanakan atas masukan baru dari pengguna. Dan yang lebih mendasar, DevOps mampu memperkecil gap antara pengembang dengan pengguna aplikasi SPMI-PT. Melalui metode DevOps, pengembangan dan pengoperasian memiliki keterhubungan sebagai timbal balik antara pengembangan dengan pengoperasian maupun sebaliknya.

## UCAPAN TERIMA KASIH

Terimakasih kepada DPRM Ristekdikti 2019 yang telah mendanai penelitian Riset Terapan sehingga paper dapat ditulis. Dan juga kami ucapkan terimakasih kepada Lembaga Pengembangan Teknologi dan Sistem Informasi (LPTSI), Lembaga Pengembangan Pembelajaran dan Penjaminan Mutu (LP3M) UNSOED yang telah bekerjasama dalam penyediaan materi penelitian.

## REFERENSI

[1] Anonim, *Buku Pedoman Sistem Penjaminan Mutu Internal (SPM-PT), Jakarta: Dirjen DIKTI*.https://spmi.ristekdikti.go.id/uploads/publications/1.%20Kebijakan%20Nasional%20SPM%20Dikti.pdf.

[2] A. D. Lestari, “Kampus Disruptif - Pusat.” .

[3] R. de Feijter, “Towards the adoption of DevOps in software product organizations: A Maturity Model Approach,” *Tech. Rep.*, 2017.

[4] CaTechnology, “TechInsights Report : What Smart Businesses Know About DevOps.” p. 300, 2013.

[5] R. Wilsenach, “DevOpsCulture,” vol. 4, no. 1, pp. 75–84, 2015.

[6] L. Ellen, L. R.- Kalliosaari, S. Mäkinen, and L. E. Lwakatare, “DevOps Adoption Benefits and Challenges in Practice : A Case Study,” pp. 590 – 597, 2016.

[7] A. Taryana, I. Setiawan, A. Fadli, and E. Murdyantoro, “Pioneering the automation of lnternal quality assurance system of higher education (IQAS-HE) using DevOps approach,” *Proc. - 2017 Int. Conf. Sustain. Inf. Eng. Technol. SIET 2017*, vol. 2018-Janua, no. August 2018, pp. 259–264, 2018.

[8] J. Wettinger, U. Breitenbücher, and F. Leymann, “Standards-based DevOps Automation and Integration Using TOSCA Institute of Architecture of Application Systems Standards - based DevOps Automation and Integration Using TOSCA,” *Proc. 7th Int. Conf. Util. Cloud Comput. (UCC 2014)*, 2014.

[9] J. Wettinger, U. Breitenbücher, and F. Leymann, "DevOpSlang - Bridging the gap between development and operations," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8745 LNCS, pp. 108–122, 2014.

[10] S. Carrizo, S. Cucu, and S. Modir, *Using Liberty for DevOps, Continuous Delivery, and Deployment*. 2015.

[11] W. Workflow and T. Binz, "web extra Portable Cloud Services Using TOSCA," *IEEE Internet Comput.*, p. 16, 2012.

[12] T. Binz *et al.*, "Institute of Architecture of Application Systems TOSCA : Portable Automated Deployment and Management of Cloud Applications The original publication will be available at www.springerlink.com TOSCA: Portable Automated Deployment and Management of Cloud Ap," 2014.

[13] T. Binz, C. Fehling, F. Leymann, A. Nowak, and D. Schumm, "Formalizing the cloud through enterprise topology graphs," *Proc. - 2012 IEEE 5th Int. Conf. Cloud Comput. CLOUD 2012*, pp. 742–749, 2012.

[14] R. Barry and H. Jay, *Operation Management*. New Jersey: Prentice Hall, 2005.

[15] A. Taryana, B. Wijayanto, and N. UBayashi, "Model-Driven Development : Fase Awal Verifikasi Model Design Rekam Medis Elektronis Menggunakan Graf Lengkap," *JMP FMIPA Unsoed*, vol. 6, pp. 53–64, 2014.