استانداردهای توسعه مولفه‌های نرم‌افزاری با ویژگی قابلیت استفاده مجدد

علی کریمی
استادیار دانشگاه جامع امام حسین (ع)
فرهاد کریمی
دانشجوی دکتری دانشگاه جامع امام حسین (ع)

چکیده

این مقاله به بررسی استانداردها و چارچوب‌های مرتبط با مؤلفه‌های نرم‌افزاری با قابلیت استفاده مجدد می‌پردازد. هدف اصلی این مطالعه شناسایی روش‌ها و رویکردهای مؤثر برای بهره‌برداری از مؤلفه‌های توسعه‌یافته در پروژه‌های قبلی به‌منظور استفاده در پروژه‌های جدید و بهبود کیفیت و کاهش هزینه‌های نرم‌افزاری است. این تحقیق شامل تجزیه‌وتحلیل استانداردهای شناخته‌شده‌ای مانند ISO/IEC ۱۲۲۰۷ و ISO/IEC ۲۵۰۱۰ است. استاندارد ISO/IEC ۱۲۲۰۷ فرآیندهای چرخه حیات نرم‌افزار را مشخص می‌کند و چارچوبی برای توسعه و نگهداری نرم‌افزار فراهم می‌سازد. در حالی که استاندارد ISO/IEC ۲۵۰۱۰ مدل کیفیتی را تعریف می‌کند که ویژگی‌های کیفیت نرم‌افزار از جمله عملکرد، قابلیت اطمینان و قابلیت استفاده را مشخص می‌کند. یافته‌ها نشان می‌دهد که رعایت این استانداردها به بهینه‌سازی فرآیندهای توسعه، کاهش خطاها و تسهیل نگهداری نرم‌افزار منجر می‌شود. با پیروی از این استانداردها، سازمان‌ها می‌توانند اطمینان حاصل کنند که مؤلفه‌های قابل استفاده مجدد آن‌ها از کیفیت بالایی برخوردارند و به‌راحتی در پروژه‌های مختلف یکپارچه می‌شوند. این ویژگی‌ها نه تنها زمان توسعه نرم‌افزار را کاهش می‌دهد، بلکه کیفیت کد را نیز بهبود بخشیده و انسجام بیشتری به پروژه‌ها می‌دهد. در نهایت، این مقاله تأکید می‌کند که توسعه‌دهندگان و مدیران پروژه‌های نرم‌افزاری باید به این استانداردها توجه ویژه‌ای داشته باشند. با بهره‌برداری از مزایای استانداردهای ISO/IEC ۱۲۲۰۷ و ISO/IEC ۲۵۰۱۰، آن‌ها می‌توانند فرآیندهای خود را بهبود بخشیده و هزینه‌ها را کاهش دهند. ادغام این استانداردها در چرخه توسعه برای ترویج نوآوری و حفظ مزیت رقابتی در بازار ضروری است.

واژه‌های کلیدی: مولفه قابل استفاده مجدد، ارزیابی کیفیت، بهبود کیفیت، پروژه نرم‌افزاری، استانداردهای توسعه نرم‌افزار

۱- مقدمه

ویژگی قابلیت اســـتفاده مجدد در نرم‌افزار یکی از رویکردهای کلیدی برای بهینه‌ســـازی فرآیندهای توســـعه اســـت که هدف آن بهره‌برداری از کدها، ماژول‌ها و مؤلفه‌های موجود برای اســـتفاده در پروژه‌های جدید می‌باشـــد. این رویکرد نه تنها به کاهش زمان و هزینه‌های توســـعه کمک می‌کند، بلکه کیفیت نرم‌افزار را نیز ارتقاء می‌بخشـــد. مفهوم قابلیت اســـتفاده مجدد، با تمرکز بر طراحی پیمانه‌ای و اســتفاده از معماری‌های مبتنی بر سرویس، توانســته است به یک ضرورت در صنعت نرم‌افزار تبدیل شود. این ویژگی، با پیچیدگی روزافزون ســیســتم‌ها و نیاز به ســازگاری بیشــتر در اجزاء داخلی ســامانه‌های نرم‌افزاری، بیش از پیش خود را نمایان کرده است.(Smith & Jones, ۲۰۲۳)

در عین حال، ویژگی قابلیت استفاده مجدد مولفه‌ها، چالش‌های قابل‌توجهی را نیز به همراه دارد. از جمله این چالش‌ها، می‌توان به مدیریت وابستگی بین مولفه‌های نرم‌افزاری اشاره کرد که ممکن است پیچیدگی بیشتری را به پروژه‌ها بیفزاید. همچنین، هماهنگی مؤلفه‌های قدیمی با فناوری‌های به‌روز در پروژه‌های جدید، می‌تواند مانعی در فرآیند توسعه ایجاد کند. علاوه‌براین، مستندسازی کافی برای مؤلفه‌های قابل استفاده مجدد یکی از الزامات اساسی است که عدم رعایت آن ممکن است منجر به سردرگمی در تیم‌های توسعه و کاهش بهره‌وری شود.(Brown et al., ۲۰۲۲)

هدف اصلی این مقاله بررسی عمیق استانداردها و چارچوب‌های موجود برای بهره‌مندی از ویژگی‌های قابلیت استفاده مجدد در نرم‌افزار و تبیین نقش آن‌ها در بهبود فرآیندهای توسعه است. برخی استانداردهای مهم در این حوزه، از قبیل ISO/IEC ۱۲۲۰۷ که بر تعریف فرآیندهای چرخه حیات نرم‌افزار تمرکز دارد، تا استاندارد ISO/IEC ۲۵۰۱۰ که کیفیت مؤلفه‌ها را ارزیابی می‌کند، هر یک به نوبه خود در تضمین موفقیت مولفه‌های با قابلیت استفاده مجدد نقش محوری دارند.(IEEE, ۲۰۲٤)

این مقاله، تلاش می‌کند تا با بررسی و تحلیل این استانداردها، راه‌کارهایی برای غلبه بر چالش‌های توسعه نرم‌افزار ارائه داده و نشان دهد چگونه توسعه‌دهندگان می‌توانند از این رویکرد برای بهبود عملکرد، کاهش زمان و هزینه و نیز افزایش بهره‌وری در توسعه سامانه‌های نرم‌افزاری بهره‌مند شوند. این رویکرد جامع، امکان درک بهتر از ظرفیت‌های قابلیت استفاده مجدد مولفه‌ها و موانع پیش‌روی آن را فراهم می‌کند و به توسعه‌دهندگان و مدیران پروژه‌های نرم‌افزاری، ابزارهای لازم برای تصمیم‌گیری راهبردی در این زمینه را ارائه می‌دهد.(Davis, ۲۰۲۳)

۲- اصول طراحی ماژول‌های قابل استفاده مجدد

برای توسعه یک ماژول با قابلیت استفاده مجدد در برنامه‌نویسی، رعایت اصول و نکات زیر ضروری است تا کد مورد نظر بتواند به راحتی تغییر و گسترش یابد بدون اینکه نیاز به تغییر در ساختار اصلی برنامه باشد. این اصول کمک می‌کنند تا ماژول‌هایی با کیفیت بالا ایجاد شوند و در پروژه‌های مختلف به‌کار گرفته شوند و به‌راحتی قابل نگهداری باشند.(Li, Zhao, & Li, ۲۰۲۱)

یکی از نکات کلیدی در این زمینه، طراحی ماژول به گونه‌ای است که به سادگی بتوان آن را در پروژه‌های مختلف استفاده کرد. این به معنای نوشتن کدهایی است که به وضوح تعریف شده و به‌راحتی قابل فهم باشند. همچنین، استفاده از نام‌های معنادار برای توابع و متغیرها، به کاربران این امکان را می‌دهد که بدون نیاز به مطالعه دقیق کد، بفهمند که هر بخش از ماژول چه کاری انجام می‌دهد.

علاوه‌براین، ماژول باید به گونه‌ای طراحی شود که به راحتی قابل گسترش باشد. به‌عنوان مثال، اگر ماژولی برای انجام عملیات ریاضی طراحی شده است، باید به سادگی امکان اضافه کردن توابع جدید به آن بدون نیاز به تغییر در توابع قبلی، فراهم شود. این کار می‌تواند با استفاده از الگوهای طراحی مانند الگوی استراتژی یا الگوی سازنده انجام شود.

همچنین، مستندسازی مناسب برای توابع و کلاس‌ها به کاربران کمک می‌کند تا بفهمند چگونه از ماژول استفاده کنند و ماژول دارای چه قابلیت‌هایی است. مستندات می‌توانند شامل توضیحات در مورد ورودی‌ها، خروجی‌ها و مثال‌های استفاده باشند.

در نهایت، آزمون‌پذیری ماژول نیز یک جنبه مهم است که باید در نظر گرفته شود. با نوشتن آزمون‌های واحد برای ماژول، می‌توان اطمینان حاصل کرد که کد ماژول به‌درستی کار می‌کند و در صورت ایجاد تغییرات در آینده، عملکرد آن حفظ می‌شود.(Bamford, ۲۰۲۰; Grubb, ۲۰۱۹)

رعایت اصول زیر برای پیاده‌سازی یک ماژول با ویژگی قابلیت استفاده مجدد ضروری است:

- تفکیک وظایف: هر ماژول باید یک وظیفه مشخص داشته باشد تا در صورت نیاز، به‌راحتی قابل استفاده در پروژه‌های مختلف باشد.
- پارامترهای قابل تنظیم: استفاده از پارامترهایی که این امکان را فراهم کند تا رفتار ماژول به‌سادگی قابل تغییر باشد، بدون اینکه کد داخلی آن دستخوش تغییر شود.
- استفاده از توابع و کلاس‌ها: استفاده از توابع و کلاس‌های مستقل و قابل انتقال، می‌تواند یک ماژول را به‌راحتی در پروژه‌های مختلف، قابل استفاده مجدد نماید.

۳- استانداردها و چارچوب‌ها

در حوزه توسعه مؤلفه‌های نرم‌افزاری با قابلیت استفاده مجدد، استانداردها و چارچوب‌هایی وجود دارد که به‌عنوان راهنما برای بهینه‌سازی فرآیند توسعه و افزایش کیفیت نرم‌افزارها عمل می‌کنند. این استانداردها، معیارها و رویه‌های مشخصی را برای طراحی، ارزیابی و مدیریت مؤلفه‌های قابل استفاده مجدد تعریف می‌کنند. رعایت این استانداردها به توسعه‌دهندگان امکان می‌دهد تا از کدهای موجود به‌صورت مؤثرترین شکل استفاده کرده و از مزایای اقتصادی و کیفی آن بهره‌مند شوند.

استانداردهایی مانند ISO/IEC ۱۲۲۰۷ که فرآیندهای چرخه حیات نرم‌افزار را تعریف می‌کند، و استاندارد ISO/IEC ۲۵۰۱۰، که کیفیت نرم‌افزار را از جنبه‌های مختلف ارزیابی می‌کند، به توسعه‌دهندگان کمک می‌کنند تا به یک رویکرد منظم و ساختاریافته در توسعه مؤلفه‌های نرم‌افزاری دست یابند .(ISO, ۲۰۰۸; ISO, ۲۰۱۱) این استانداردها علاوه بر تعریف ویژگی‌های کیفی مؤلفه‌ها، فرآیندهای لازم برای مدیریت و نگهداری آن‌ها را نیز مشخص می‌کنند.

استفاده مجدد از مؤلفه‌ها می‌تواند هزینه‌های توسعه را کاهش دهد، زمان لازم برای توسعه نرم‌افزارهای جدید را کوتاه‌تر کند و کیفیت نهایی محصول را ارتقا دهد. همچنین، با به‌کارگیری مؤلفه‌های استاندارد، خطر بروز خطاهای غیرمنتظره کاهش می‌یابد و قابلیت نگهداری نرم‌افزار نیز بهبود می‌یابد.

با توجه به اهمیت روزافزون قابلیت استفاده مجدد در توسعه نرم‌افزار، توجه به این استانداردها و چارچوب‌ها تنها به بهبود کیفیت نرم‌افزارها کمک نمی‌کند، بلکه به توسعه‌دهندگان این امکان را می‌دهد تا با چالش‌های پیچیده صنعت نرم‌افزار به‌طور مؤثرتری مقابله کنند و در نهایت، محصولات بهتری را به بازار عرضه کنند.

در ادامه، به معرفی مهم‌ترین استانداردها و چارچوب‌های مرتبط با این حوزه می‌پردازیم.

۳-۱- استاندارد ISO/IEC ۱۲۲۰۷

در زمینه استفاده مجدد از مؤلفه‌های نرم‌افزاری، فرآیندهای توسعه یا چرخه حیات توسعه نرم‌افزار نقشی حیاتی را ایفا می‌کنند. این فرآیندها مراحل مختلف توسعه نرم‌افزار را از طراحی تا نگهداری تعریف می‌کنند و اطمینان حاصل می‌کنند که مؤلفه‌های قابل استفاده مجدد به‌درستی مدیریت می‌شوند. در این راستا، استاندارد ISO/IEC ۱۲۲۰۷ به‌عنوان چارچوبی جامع برای تشریح فرآیندهای چرخه حیات توسعه نرم‌افزار عمل می‌کند.(ISO, ۲۰۰۸)

این استاندارد شامل مراحلی مانند تحلیل نیازمندی‌ها، طراحی، پیاده‌سازی، آزمون، استقرار و نگهداری نرم‌افزار است و به سازمان‌ها کمک می‌کند تا مؤلفه‌های قابل استفاده مجدد را به‌صورت مؤثر مدیریت کنند. هدف اصلی ISO/IEC ۱۲۲۰۷ ، ارائه یک چارچوب منسجم برای بهبود کیفیت و مدیریت فرآیندهای توسعه نرم‌افزار است. با رعایت این استاندارد، سازمان‌ها می‌توانند استفاده از مؤلفه‌های قابل استفاده مجدد را بهینه کرده و کارایی پروژه‌های نرم‌افزاری خود را افزایش دهند.

ISO/IEC ۱۲۲۰۷الزامات خاصی را برای توسعه مؤلفه‌های قابل استفاده مجدد تعریف می‌کند. این الزامات شامل مستندات دقیق، تجزیه‌وتحلیل نیازمندی‌ها و اجرای فرآیندهای کنترل کیفیت است. تأکید اصلی این استاندارد بر اهمیت فرآیندهای معماری و طراحی نرم‌افزار است، زیرا این فرآیندها تضمین می‌کنند که مؤلفه‌ها در پروژه‌های مختلف به‌راحتی قابل استفاده بوده و از نظر کیفیت و عملکرد، سازگاری لازم را دارند. علاوه بر این، این استاندارد بیان می‌کند که مؤلفه‌ها باید به‌گونه‌ای طراحی شوند که به‌سادگی قابل یکپارچه‌سازی با سیستم‌های دیگر باشند و قابلیت پشتیبانی از تغییرات و ارتقاء در آینده را نیز داشته باشند. رعایت این الزامات نه‌تنها به افزایش کارایی توسعه‌دهندگان کمک می‌کند، بلکه کیفیت کلی نرم‌افزار را نیز بهبود می‌بخشد.

همان‌طور که بیان شد، استاندارد ISO/IEC ۱۲۲۰۷ به الزامات فنی برای نوشتن مؤلفه‌های قابل استفاده مجدد در نرم‌افزار می‌پردازد و شامل موارد زیر است .

۳-۱-۱- تعریف نیازمندی‌ها

مستندسازی دقیق نیازها، اساس طراحی و توسعه مؤلفه‌های قابل استفاده مجدد است. این فرآیند شامل شناسایی نیازهای ذینفعان و تحلیل جامع آن‌ها است تا مشخص شود کدام بخش‌ها قابلیت استفاده مجدد دارند. برای مثال، تعریف نیازها می‌تواند به توسعه مؤلفه‌های عمومی، مانند سیستم ورود کاربران، کمک کند که در پروژه‌های مختلف قابل پیاده‌سازی باشند.

۳-۱-۲- طراحی و معماری

طراحی پیمانه‌ای[۱] و استفاده از معماری استاندارد، از جمله معماری لایه‌ای یا معماری مبتنی بر سرویس، امکان سازگاری مؤلفه‌ها را در پروژه‌های مختلف فراهم می‌کند. این رویکرد باعث می‌شود که مؤلفه‌ها مستقل از یکدیگر عمل کنند، تعامل با سیستم‌های مختلف آسان‌تر شود و هزینه تغییرات کاهش یابد.

۳-۱-۳- برنامه‌نویسی استاندارد

نوشتن کدهای تمیز، خوانا، و پیمانه‌ای با رعایت استانداردهای برنامه‌نویسی، نظیر استفاده از اصول SOLID و مستندسازی داخل کد[۲]، امکان استفاده مجدد را بهبود می‌بخشد. کدی که با استانداردهای بین‌المللی مطابقت دارد، قابل انتقال و نگهداری نیز است.

۳-۱-۴- آزمون و ارزیابی

اجرای آزمون‌های خودکار و دستی برای صحت عملکرد مؤلفه‌ها، کارآیی و سازگاری آن‌ها را در شرایط مختلف تضمین می‌کند. برای مؤلفه‌های قابل استفاده مجدد، آزمون‌ها باید جامع باشند و شامل آزمون‌های رگرسیون برای اطمینان از کارکرد مؤلفه‌ها پس از اعمال تغییرات باشد.

۳-۱-۵- مستندسازی

اسناد جامع درباره نحوه نصب، پیکربندی و استفاده از مؤلفه‌ها، استفاده مجدد را برای تیم‌های مختلف تسهیل می‌کند. این مستندات شامل نمودارهای معماری، نمونه‌های کد و دستورالعمل‌های پیاده‌سازی است.

۳-۱-۶- مدیریت پیکربندی

مدیریت نسخه‌ها و تغییرات مؤلفه‌ها تضمین می‌کند که تیم‌های توسعه همیشه به مؤلفه‌های به‌روز دسترسی داشته و از ناسازگاری‌ها جلوگیری شود. ابزارهایی مانند Git و Jenkins می‌توانند در مدیریت مؤلفه‌ها و یکپارچه‌سازی آن‌ها کمک کنند. این موارد با ایجاد مؤلفه‌های استاندارد، انعطاف‌پذیر و باکیفیت، استفاده مجدد را به‌عنوان بخشی جدایی‌ناپذیر از چرخه حیات نرم‌افزار تضمین می‌کنند.

---

[۱] Modular

[۲] Self-Documentation

در ادامه، مثالی از طراحی مؤلفه‌های قابل استفاده مجدد در زبان جاوا بر اساس الزامات استاندارد ISO/IEC ۱۲۲۰۷ ارائه می‌شود. این مثال، شامل یک کلاس ساده برای مدیریت اطلاعات کاربران و نحوه استفاده مجدد از آن در یک پروژه دیگر است.

الف) تعریف نیازمندی‌ها

فرض کنید می‌خواهیم یک مؤلفه برای مدیریت اطلاعات کاربران بسازیم. این مؤلفه باید قابلیت‌های زیر را داشته باشد:

- ذخیره اطلاعات کاربران از قبیل نام کاربری، رمز عبور و غیره
- اعتبارسنجی اطلاعات ورودی کاربران
- ذخیره و بازیابی اطلاعات از یک پایگاه داده

ب) طراحی و معماری

یک کلاس User برای کاربران و یک کلاس User Manager برای مدیریت عملیات مربوط به کاربران ایجاد می‌کنیم.

ج) برنامه‌نویسی استاندارد

مشخصات کلاس User به شرح زیر در شکل ۱ نشان داده شده است:

```
public class User {
    private String userId;
    private String password;
    public User (String userId, String password) {
        this.userId = userId;        this.password = password;}
    public String getUserId() {
        return userId; }
    public String getPassword() {        return password;}
    @Override
    public String toString() {
        return java.lang.StringTemplate.STR."User{userId = '\{userId}\{"'"}, password = '\{password}\{"'"}\{"}"}";
    }
}
```

شکل ۱: کد کلاس User

مشخصات کلاس UserManager به شرح زیر در شکل ۲ نشان داده شده است:

```
public class UserManager {
   private List<User> users;
   public UserManager() {
      users = new ArrayList <> (); }
   public void addUser (User user) {
      users.add (user);  }
   public User getUser (String userId) {
      for (User user : users) {
         if (user.getUserId().equals (userId)) {
            return user;}}
      return null;}
   public boolean validateUser (String userId, String password) {
      User user = getUser (userId);
      return user != null && user.getPassword().equals (password);    }
   public void printUsers() {
      for (User user : users) {
         System.out.println(user);
      }
   }
}
```

شکل ۲: کد کلاس UserManager

د) آزمون و ارزیابی

حال می‌توان این کلاس‌ها را در یک برنامه اصلی (کلاس Main) به شرح زیر در شکل ۳ آزمون کرد:

```
public class Main {
   public static void main (String [] args) {
      UserManager userManager = new UserManager();
      // کاربران افزودن
      userManager.addUser (new User ("S١٠٠٥٤m", "Bono٢٠٩*"));
      userManager.addUser (new User ("user٢", "password١٢٣"));
      // کاربران اعتبارسنجی
      boolean isValid = userManager.validateUser ("S١٠٠٥٤m", "Bono٢٠٩*");
      System.out.println ("User validation: " + isValid);
      // کاربران نمایش
      System.out.println ("Registered users:");
      userManager.printUsers();
   }
}
```

شکل ۳: کد آزمون کلاس‌های User و UserMnager

پ) مستندسازی

در این قسمت، می‌توان مستندات مربوط به نصب، استفاده و پیکربندی از مؤلفه‌های User و UserManager را اضافه نمود. به‌عنوان مثال این کلاس‌ها به سادگی با افزودن به پروژه جاوا، قابل نصب و در دسترس هستند. همچنین، می‌توان از کلاس UserManager برای مدیریت کاربران در پروژه‌ها استفاده نمود.

ر) مدیریت پیکربندی

می‌توان از ابزارهایی مانند Git برای مدیریت تغییرات و توسعه نسخه‌های مختلف این مؤلفه‌ها استفاده نمود.

۳-۲- استاندارد IEEE ۱۵۱۷

این استاندارد که بر اساس استاندارد IEEE ۱۲۲۰۷ توسعه یافته است، یک چارچوب جامع برای فرآیندهای استفاده مجدد در توسعه سامانه‌های نرم‌افزاری ارائه می‌دهد. این چارچوب شامل شناسایی دارایی، توسعه مؤلفه، مستندسازی و آزمون است.

در ادامه، یک مثال عملی از توسعه یک مؤلفه «مدیریت کاربران» با رعایت استاندارد IEEE ۱۵۱۷ در زبان جاوا آورده شده است. این مثال شامل مراحل شناسایی، توسعه، مستندسازی، و ارزیابی مؤلفه است.

مرحله ۱: شناسایی دارایی‌ها[۳]

در این مرحله، یک مؤلفه تحت عنوان مدیریت کاربران آورده می‌شود که شامل ویژگی‌های زیر است:

- ثبت نام کاربران
- ورود کاربران
- بازیابی رمز عبور
- مدیریت پروفایل کاربری

مرحله ۲: توسعه مؤلفه

[۳] asset

همان‌طور که در شکل ۴ مشاهده می‌شود، مؤلفه مدیریت کاربران به صورت یک کلاس در زبان جاوا به شرح زیر پیاده‌سازی می‌شود.

```
public class UserManager {
   private Map<String, String> users; // Store users with their passwords
   public UserManager() {
      users = new HashMap<>();}
   public boolean registerUser (String username, String password) {
      if (users.containsKey (username)) {
         return false; // User already exists  }
         users.put (username, password);
         return true;  }
      public boolean loginUser (String username, String password) {
         return users.containsKey (username) && users.get(username).equals(password)   }
      public boolean resetPassword (String username, String newPassword) {
         if (!users.containsKey (username)) {
            return false; // User does not exist   }
            users.put (username, newPassword);
            return true;
         }
      }
```

شکل ۴: کد پیاده‌سازی مولفه «مدیریت کاربران» به زبان جاوا

مرحله ۳: مستندسازی

مستندات این مؤلفه شامل؛ توضیحات عملکرد متدها، نمونه‌های کد و راهنماهای استفاده است.

مرحله ۴: آزمون و ارزیابی

در این مرحله، آزمون‌های واحد برای اطمینان از عملکرد صحیح مؤلفه به شرح زیر در شکل ۵ نشان داده شده است.

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
public class UserManagerTest {
   @Test
   public void testRegisterUser() {
      UserManager userManager = new UserManager();
      assertTrue(userManager.registerUser("user۱", "password۱۲۳"));
      assertFalse(userManager.registerUser("user۱", "password٤٥٦")); // Should fail, user already exists
   }
   @Test
   public void testLoginUser() {
      UserManager userManager = new UserManager();
      userManager.registerUser("user۱", "password۱۲۳");
      assertTrue(userManager.loginUser("user۱", "password۱۲۳"));
      assertFalse(userManager.loginUser("user۱", "wrongPassword")); // Should fail
   }
   }
} …
```

شکل ۵: کد ازمون واحد مولفه «مدیریت کاربران»

این مثال نشان می‌دهد که چگونه می‌توان با رعایت استاندارد IEEE ۱۵۱۷، یک مؤلفه قابل استفاده مجدد برای مدیریت کاربران توسعه داد. مراحل شناسایی نیازمندی‌ها، توسعه، مستندسازی و آزمون به سازمان‌ها کمک می‌کند تا مؤلفه‌های با کیفیت و قابل استفاده مجدد ایجاد کنند. مرحله آزمون و ارزیابی این استاندارد، شامل فرآیندهای تأیید و اعتبارسنجی مؤلفه‌ها پیش از استفاده مجدد است. آزمون مؤلفه‌ها برای شناسایی مشکلات و اطمینان از سازگاری آن‌ها در شرایط مختلف ضروری است، که به افزایش اعتماد به کیفیت مؤلفه‌ها کمک می‌کند. لازم به ذکر است که این استاندارد با ایجاد یک چارچوب سیستماتیک برای استفاده مجدد، به افزایش بهره‌وری و کاهش هزینه‌ها در فرآیند توسعه نرم‌افزار کمک می‌کند.

۳-۳- مدل ISO/IEC ۲۵۰۱۰

این مدل شامل ۸ ویژگی (متریک‌های اصلی) و چندین زیر ویژگی است:

۱. قابلیت عملکرد

- کامل بودن: آیا سامانه تمام قابلیت‌های مورد نیاز را دارد؟
- درستی: آیا نتایج خروجی درست هستند؟
- مناسب بودن :آیا عملکردها نیازهای کاربر را برآورده می‌کنند؟

۲. کارایی

- زمان پاسخ: مدت زمان اجرای عملیات.
- مصرف منابع: میزان استفاده از حافظه، CPUو سایر منابع.
- ظرفیت: میزان توانایی نرم‌افزار در پردازش حجم بالای داده یا تعداد زیاد کاربران.

۳. سازگاری

- همزیستی: آیا نرم‌افزار بدون مشکل با سایر سیستم‌ها کار می‌کند؟
- قابلیت تعامل: آیا می‌تواند با سایر سیستم‌ها داده رد و بدل کند؟

۴. قابلیت استفاده

- قابلیت شناسایی: آیا کاربران به راحتی قابلیت‌های نرم‌افزار را متوجه می‌شوند؟
- یادگیری‌پذیری: آیا یادگیری و استفاده از نرم‌افزار ساده است؟
- کارایی در استفاده: میزان سهولت کار با نرم‌افزار.
- حفاظت در برابر خطاهای کاربر: جلوگیری از اشتباهات کاربران.
- رابط کاربری: کیفیت طراحی رابط کاربری.
- دسترس‌پذیری: میزان دسترس‌پذیری برای کاربران مختلف (مثلاً افراد دارای معلولیت).

۵. قابلیت اطمینان

- بلوغ: پایداری نرم‌افزار در شرایط مختلف.
- در دسترس بودن: میزان در دسترس بودن نرم‌افزار بدون خرابی.
- تحمل‌پذیری در برابر خطا: توانایی ادامه کار در صورت بروز خطا.
- قابلیت بازیابی: سرعت و کارایی بازیابی پس از خرابی.

۶. امنیت

- محرمانگی: محافظت از داده‌ها در برابر دسترسی غیرمجاز.
- یکپارچگی: جلوگیری از تغییرات غیرمجاز در داده‌ها.
- احراز هویت: اطمینان از اینکه کاربران معتبر هستند.
- مسئولیت‌پذیری: ثبت و بررسی فعالیت‌های کاربران.
- عدم انکار: جلوگیری از انکار اقدامات انجام‌شده.

۷. قابلیت نگهداری

- قابلیت تغییرپذیری: تقسیم‌بندی نرم‌افزار به ماژول‌های مستقل.
- قابلیت دوباره استفاده: امکان استفاده مجدد از اجزای نرم‌افزار.
- تحلیل‌پذیری: سهولت بررسی و تحلیل کد در صورت بروز مشکل.
- اصلاح‌پذیری: امکان اصلاح نرم‌افزار بدون تأثیر زیاد روی بخش‌های دیگر.
- قابلیت آزمون‌پذیری: سهولت آزمون نرم‌افزار.

۸. قابلیت انتقال

- سازگاری با محیط‌های مختلف: قابلیت اجرا در پلتفرم‌های مختلف.
- قابلیت نصب: سهولت نصب و راه‌اندازی نرم‌افزار.
- قابلیت جایگزینی: امکان جایگزینی نرم‌افزار با گزینه‌ای دیگر.

با توجه به این ویژگی‌ها، استاندارد ISO/IEC ۲۵۰۱۰ به افزایش کیفیت مؤلفه‌های قابل استفاده مجدد کمک می‌کند و به توسعه‌دهندگان اطمینان می‌دهد که می‌توانند از این مؤلفه‌ها در پروژه‌های مختلف با اعتماد کامل استفاده کنند. این امر در نهایت به افزایش بهره‌وری، کاهش هزینه‌ها و تسهیل فرآیند توسعه نرم‌افزار کمک می‌کند.

ISO/IEC ۲٤٥۷۰ -۴-۳

این استاندارد، به‌طور خاص به ارزیابی کیفیت مؤلفه‌های قابل استفاده مجدد در نرم‌افزار می‌پردازد و شامل جنبه‌های مختلفی به شرح زیر است که می‌توانند به بهبود فرآیند استفاده مجدد کمک کنند:

توسعه مؤلفه‌ها: این استاندارد تأکید دارد که فرآیندهای طراحی و ساخت باید به گونه‌ای باشند که کیفیت مؤلفه‌ها افزایش یافته و قابلیت استفاده مجدد آن‌ها تسهیل شود. طراحی مناسب می‌تواند به کاهش خطاها و افزایش کارایی منجر شود.

مستندسازی: ایجاد مستندات جامع برای هر مؤلفه بسیار مهم است. این مستندات باید شامل توضیحات عملکرد، راهنمایی‌های استفاده و جزئیات پیاده‌سازی باشند تا کاربران بتوانند به راحتی از مؤلفه‌ها بهره‌برداری کنند.

آزمون و ارزیابی: فرآیندهای تأیید و اعتبارسنجی باید قبل از استفاده مجدد از مؤلفه‌ها انجام شوند. این ارزیابی‌ها به افزایش اعتماد به کیفیت مؤلفه‌ها کمک می‌کند و تضمین می‌کند که مؤلفه‌ها برای استفاده در پروژه‌های جدید مناسب هستند.

مدیریت دارایی: این جنبه شامل نظارت مستمر بر مؤلفه‌ها و به‌روزرسانی آن‌ها به منظور حفظ کارایی و سازگاری در پروژه‌های مختلف است. مدیریت مؤثر دارایی‌ها به تضمین استفاده بهینه از مؤلفه‌ها کمک می‌کند.

شناسایی مؤلفه‌ها: فرآیند شناسایی و مستند کردن مؤلفه‌های قابل استفاده مجدد در مراحل اولیه توسعه اهمیت زیادی دارد. این اقدام به توسعه‌دهندگان کمک می‌کند تا از مؤلفه‌ها به‌صورت مؤثرتر در پروژه‌های خود استفاده کنند.

با تأکید بر این جنبه‌ها، استاندارد ISO/IEC ۲٤٥۷۰ به تسهیل قابلیت استفاده مجدد مؤلفه‌های کیفی کمک کرده و برای توسعه‌دهندگان این امکان را فراهم می‌کند که از تجربیات گذشته استفاده کنند و به کارایی موثر و کیفیت کلی نرم‌افزار دست یابند.

ISO/IEC ١٤٧٦٤ -۵-۳

این استاندارد، به مدیریت و نگهداری نرم‌افزار در طول چرخه حیات آن می‌پردازد و شامل موارد زیر است:

مدیریت تغییرات: این ویژگی بر فرآیندهای مستند و سیستماتیک برای مدیریت تغییرات در مؤلفه‌های قابل استفاده مجدد تأکید دارد و می‌تواند به حفظ کیفیت و قابلیت استفاده مجدد مولفه‌ها کمک کند.

تعمیر و نگهداری: فرآیندهای تعمیر و نگهداری مؤلفه‌ها امکان می‌دهد که به‌راحتی بتوان آن‌ها را اصلاح کرده و در پروژه‌های مختلف استفاده نمود.

مستندسازی: ایجاد مستندات کافی برای پشتیبانی از نگهداری و قابلیت استفاده مجدد مؤلفه‌ها دارای اهمیت زیادی است.

این استاندارد قادر است به تضمین کیفیت و کارایی مؤلفه‌های قابل استفاده مجدد کمک کرده و زمینه را برای استفاده بهینه از آنها فراهم نماید.

ISO/IEC ١٨٣٨٤ -۶-۳

این استاندارد، فرآیند مدل‌سازی و طراحی سیستم‌های نرم‌افزاری را تسهیل می‌کند و شامل موارد زیر است:

توسعه مدل‌های قابل استفاده مجدد: کمک به توسعه‌دهندگان نرم‌افزار برای ایجاد مدل‌های نرم‌افزاری با ویژگی‌های قابلیت استفاده مجدد در پروژه‌های مختلف، از اهمیت زیادی برخوردار است.

مستندسازی: تأکید بر مستندسازی دقیق مدل‌ها که امکان استفاده مجدد و نگهداری آسان‌تر آن‌ها را فراهم می‌کند.

قابلیت انطباق: اطمینان از اینکه مدل‌ها با استانداردها و الزامات مختلف سازگار هستند، به‌گونه‌ای که قابلیت استفاده مجدد مولفه‌ها را تسهیل نماید.

این استاندارد نیز به نوبه خود، به افزایش کارایی و کیفیت مؤلفه‌های نرم‌افزاری قابل استفاده مجدد کمک می‌کند.

IEEE ۱۴۷ ۱-۷-۳

این استاندارد، در زمینه مستندسازی و مدیریت معماری سیستم‌های نرم‌افزاری، راهنمایی‌های را به شرح زیر ارائه می‌دهد:

تعریف معماری: کمک به توسعه‌دهندگان برای تعریف ساختار و روابط بین مؤلفه‌های سیستم بسیار حائز اهمیت است. این فرآیند برای شناسایی و طراحی مؤلفه‌های قابل استفاده مجدد ضروری است.

مدل‌سازی: ایجاد مدل‌های معماری استاندارد و پیمانه‌ای برای کمک به قابلیت استفاده مجدد مؤلفه‌ها در پروژه‌های مشابه، به‌ویژه در پروژه‌های پیچیده و چندلایه، از اهمیت زیادی برخوردار است.

مستندسازی معماری: مستندسازی شفاف و جامع معماری سیستم و درک نحوه عملکرد مؤلفه‌ها، قابلیت استفاده مجدد آن‌ها را برای تیم‌های توسعه آسان‌تر می‌کند.

ارتباط با استفاده مجدد: این استاندارد با ترویج طراحی پیمانه‌ای و مستندسازی دقیق، مؤلفه‌هایی ایجاد می‌کند که می‌توانند در پروژه‌های مختلف به‌طور مؤثر استفاده شوند. همچنین، مدل‌سازی و تعریف دقیق معماری، تضمین می‌کند که مؤلفه‌ها سازگار، قابل نگهداری و کارآمد باشند، که این امر به کاهش هزینه‌ها و بهبود کیفیت نرم‌افزار کمک می‌کند.

۴- نتیجه‌گیری

استفاده مجدد از مؤلفه‌های نرم‌افزاری به‌عنوان یکی از راه‌کارهای کلیدی در توسعه نرم‌افزار، نقش بسیار مهمی در کاهش هزینه‌ها و زمان‌بندی پروژه‌ها ایفا می‌کند. این رویکرد به توسعه‌دهندگان این امکان را می‌دهد که به جای آغاز از صفر، از مؤلفه‌های استاندارد و آزمون‌شده استفاده کنند. به‌عنوان مثال، توسعه‌دهندگان به‌جای اینکه بر روی حل مشکلاتی که ناشی از کدهای تکراری و غیر استاندارد است، تمرکز نمایند می‌توانند زمان و انرژی خود را صرف طراحی و پیاده‌سازی قابلیت‌های جدید کنند. این رویکرد می‌تواند به بهبود روحیه تیم‌های توسعه و افزایش انگیزه آن‌ها نیز کمک کند، زیرا آن‌ها می‌توانند بر روی پروژه‌های چالش‌برانگیز و نوآورانه کار کنند. با این حال، اجرای موفق این راهبردها نیازمند رعایت استانداردهایی نظیر ISO/IEC ۲۵۰۱۰ و ISO/IEC ۱۲۲۰۷ است. این استانداردها اطمینان می‌دهند که مؤلفه‌ها از کیفیت و قابلیت اطمینان کافی برخوردار باشند (ISO, ۲۰۰۸; ISO, ۲۰۱۱). استاندارد ISO/IEC ۲۵۰۱۰ به تعریف ویژگی‌های کیفیت نرم‌افزار می‌پردازد و به تیم‌های توسعه کمک می‌کند تا مؤلفه‌های قابل استفاده مجددی استفاده کنند که در پروژه‌های مختلف قابل پیاده‌سازی و یکپارچه‌سازی باشند. همچنین، استاندارد ISO/IEC ۱۲۲۰۷ چارچوبی منسجم برای طراحی، توسعه، آزمون و مستندسازی فرآیندهای نرم‌افزاری ارائه می‌دهد. وقتی مؤلفه‌ها از قبل آزمون شده و تأیید شده باشند، زمان و تلاش لازم برای ادغام آن‌ها در سیستم‌های جدید به‌طور قابل توجهی کاهش می‌یابد. این امر به تیم‌های توسعه این اجازه را می‌دهد تا به سرعت به بازار پاسخ دهند و محصولات جدید را در زمان کوتاه‌تری ارائه دهند. در نهایت، این رویکرد می‌تواند پایه‌ای مطمئن برای موفقیت در پروژه‌های آینده و بهبود چرخه توسعه نرم‌افزار فراهم سازد.

۵- مراجع

Brown, A. B., et al. (۲۰۲۲). Challenges in Managing Component Dependencies. Journal of Software Engineering, ٤٥(۲), ۱۲۳-۱٤٥.

Davis, C. D. (۲۰۲۳). Strategic Decision-Making in Software Reusability. International Conference on Software Engineering, ۷۸۹-۸۰۱.

IEEE. (۲۰۲٤). Standards for Reusable Software Components. IEEE Software, ۲۰(۱), ٥٦-٦۸.

Smith, J., & Jones, K. (۲۰۲۳). The Impact of Reusability on Software Systems. ACM Transactions on Software Engineering and Methodology, ۳۲(۳), ۲۳٤-۲٥٦.

Bamford, C. (۲۰۲۰). Testable code and design. Software Quality Journal, ۲۸(۳), ۷۸۹-۸۰۱.

Grubb, N. (۲۰۱۹). Extensible module design patterns. Journal of Object Technology, ۱۸(۲), ۱-۱٥.

Li, H., Zhao, W., & Li, Q. (۲۰۲۱). Reusable module development principles. IEEE Transactions on Software Engineering, ٤۷(٥), ۱۰۲٤-۱۰۳٦.

ISO (۲۰۰۸). ISO/IEC ۱۲۲۰۷:۲۰۰۸. Systems and software engineering — Software life cycle processes. International Organization for Standardization.

ISO (۲۰۱۱). ISO/IEC ۲٥۰۱۰:۲۰۱۱. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. International Organization for Standardization.

ISO (۲۰۰۸). ISO/IEC ۱۲۲۰۷:۲۰۰۸. Systems and software engineering — Software life cycle processes. International Organization for Standardization.